B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Lab Record

Object-Oriented Modeling – 23CS5PCOOM

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

DAIVYA PRIYANKKUMAR SHAH

1BM23CS084

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560019
August 2025-December 2025

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Modeling(23CS5PCOOM) laboratory has been carried out by DAIVYA PRIYANKKUMAR SHAH(1BM23CS084) during the 5th Semester August 2025-December 2025.

Signature of the Faculty Incharge:

Name of your Batch Incharge and designation: Sonika Sharma D, Assistant Professor

Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System	3-9
2. Credit Card Processing	10-16
3. Library Management System	17-23
4. Stock Maintenance System	24-30
5. Passport Automation System	31-37

GitHub Link: https://github.com/daivya17/OOM_Lab

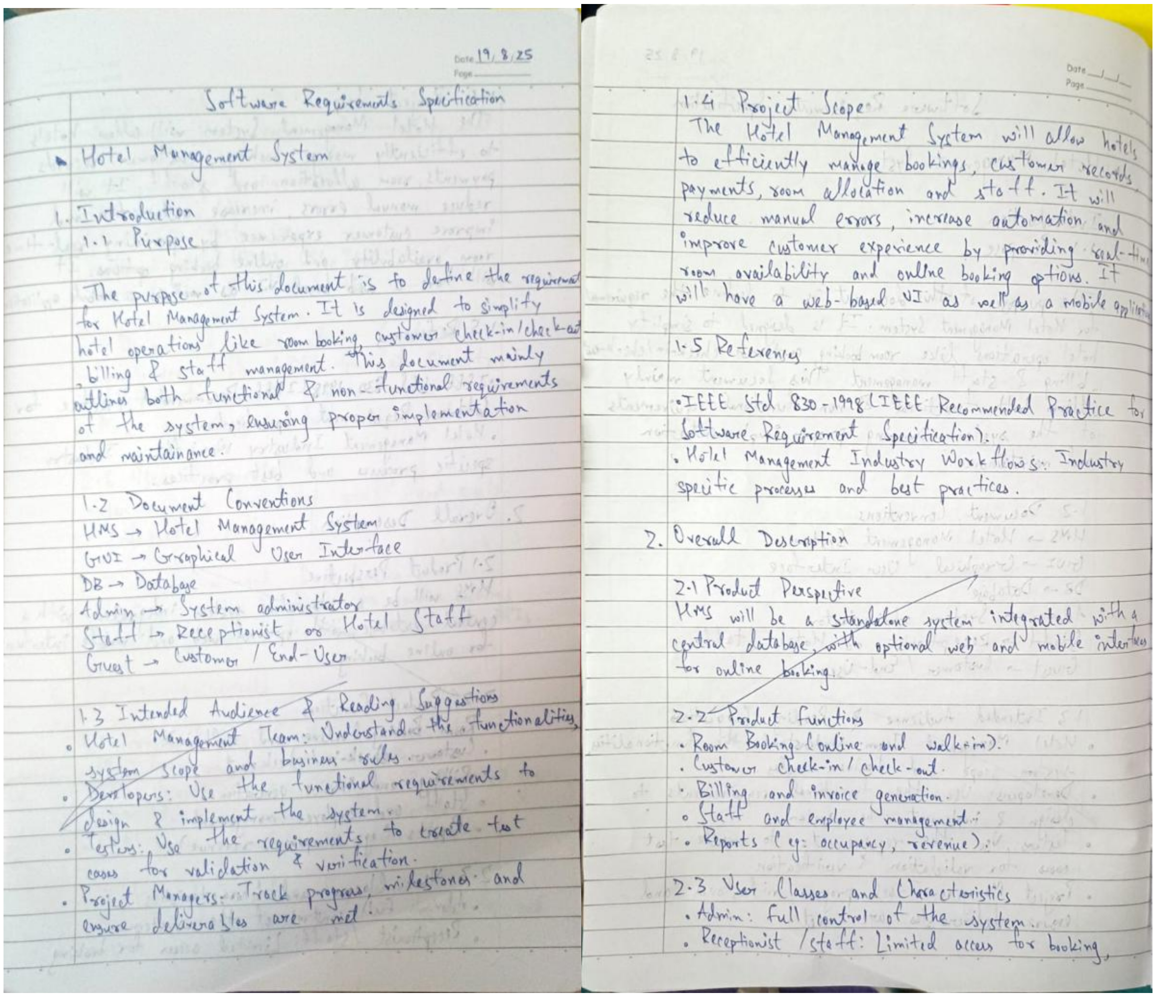
1. Hotel Management System

Problem Statement

A hotel needs a computerized system to manage its day-to-day operations efficiently. The current manual process used for managing room bookings, guest check-in/check-out, billing, and staff coordination is time-consuming, error-prone, and difficult to track. To overcome these problems, the hotel requires a Hotel Management System that automates major operational activities and improves service quality.

The system should allow customers to search for available rooms based on type, price, and date. It should support online and offline room reservations. When guests arrive, the receptionist should be able to check them in, assign rooms, and maintain guest records. Upon checkout, the system must generate accurate bills that include room charges, additional services (restaurant, laundry, spa, etc.), and applicable taxes.

SRS-Software Requirements Specification



Date 19. 8. 25

Software Requirements Specification

- Hotel Management System

1. Introduction

1.1. Purpose

The purpose of this document is to define the requirement for Hotel Management System. It is designed to simplify hotel operations like room booking, customer check-in/check-out, billing & staff management. This document mainly outlines both functional & non-functional requirements of the system, ensuring proper implementation and maintainance.

1.2 Document Conventions

HMS → Hotel Management System
GUI → Graphical User Interface
DB → Database
Admin → System administrator
Staff → Receptionist or Hotel Staff
Guest → Customer / End-User

1.3 Intended Audience & Reading Suggestions

- Hotel Management Team: Understand the functionalities, system scope and business rules.
- Developers: Use the functional requirements to design & implement the system.
- Testers: Use the requirements to create test cases for validation & verification.
- Project Managers: Track progress, milestones and ensure deliverables are met.

Date _/_/_
Page _

1.4 Project Scope

The Hotel Management System will allow hotels to efficiently manage bookings, customer records, payments, room allocation and staff. It will reduce manual errors, increase automation and improve customer experience by providing real-time room availability and online booking options. It will have a web-based UI as well as a mobile application

1.5 References

- IEEE Std 830-1998 (IEEE Recommended Practice for Software Requirement Specification).
- Hotel Management Industry Workflows: Industry specific processes and best practices.

2. Overall Description

2.1 Product Perspective

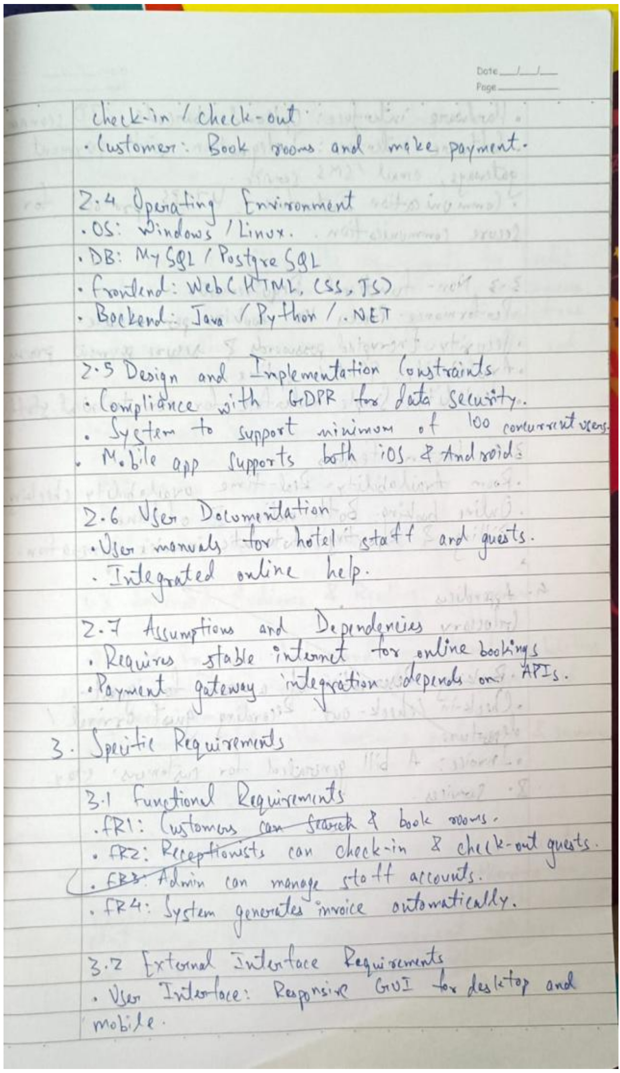
HMS will be a standalone system integrated with a central database, with optional web and mobile interfaces for online booking.

2.2 Product functions

- Room Booking (online and walk-in).
- Customer check-in / check-out.
- Billing and invoice generation.
- Staff and employee management.
- Reports (eg: occupancy, revenue).

2.3 User Classes and Characteristics

- Admin: full control of the system.
- Receptionist / staff: Limited access for booking,



Date __/__/__
Page ____

check-in / check-out.
- Customer: Book rooms and make payment.

2.4 Operating Environment
- OS: Windows / Linux.
- DB: MySQL / Postgre SQL
- frontend: Web (HTML, CSS, JS)
- Backend: Java / Python / .NET

2.5 Design and Implementation Constraints
- Compliance with GDPR for data security.
- System to support minimum of 100 concurrent users.
- Mobile app supports both iOS & Android.

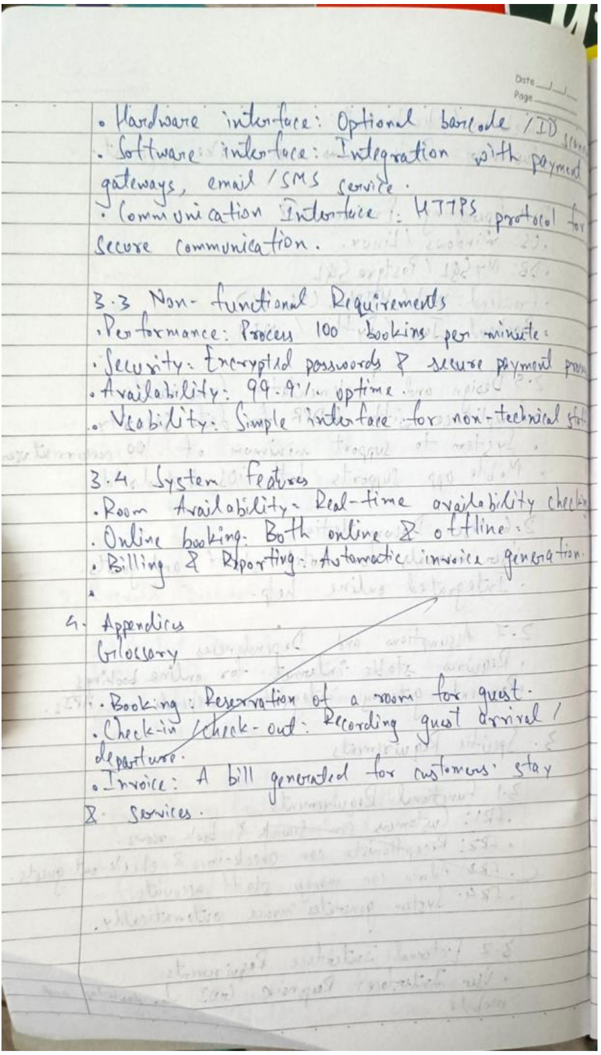
2.6 User Documentation
- User manuals for hotel staff and guests.
- Integrated online help.

2.7 Assumptions and Dependencies
- Requires stable internet for online bookings
- Payment gateway integration depends on APIs.

3. Specific Requirements

3.1 Functional Requirements
- FR1: Customers can search & book rooms.
- FR2: Receptionists can check-in & check-out guests.
- FR3: Admin can manage staff accounts.
- FR4: System generates invoice automatically.

3.2 External Interface Requirements
- User Interface: Responsive GUI for desktop and mobile.



Date __/__/__
Page ____

- Hardware interface: Optional barcode / ID scan
- Software interface: Integration with payment gateways, email / SMS service.
- Communication Interface: HTTPS protocol for secure communication.

3.3 Non-functional Requirements
- Performance: Process 100 bookins per minute.
- Security: Encrypted passwords & secure payment proc
- Availability: 99.9% uptime.
- Usability: Simple interface for non-technical staff

3.4 System features
- Room Availability: Real-time availability checking
- Online booking: Both online & offline.
- Billing & Reporting: Automactic invoice generation.
-

4. Appendices
Glossary

- Booking: Reservation of a room for guest.
- Check-in / check-out: Recording guest arrival / departure.
- Invoice: A bill generated for customers' stay & services.

Class Diagram

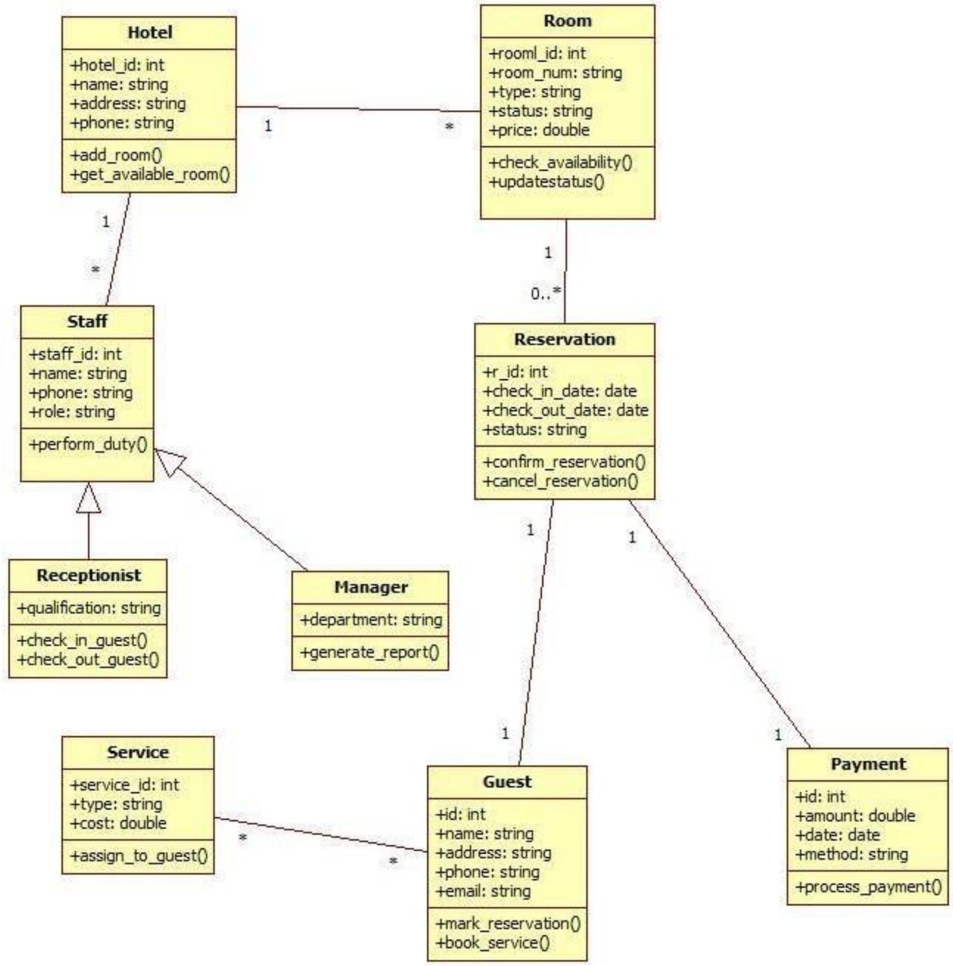


Figure 1.1: Class Diagram for Hotel Management System

A hotel management system keeps track of hotels, their rooms, and the staff who operate them. Each hotel has multiple rooms, and every room stores details such as room number, type, status, and price. Guests can make reservations for specific rooms, and each reservation contains check-in and check-out dates along with its status. Guests can also book additional services offered by the hotel, such as room service or laundry, and all payments made by guests are recorded with details like amount, date, and method. Staff members, including receptionists and managers, perform various duties such as checking guests in or generating reports, and every staff member belongs to a hotel. The system ensures that room availability is updated through reservations, guests can manage their bookings, and staff can perform their roles efficiently.

State Diagram

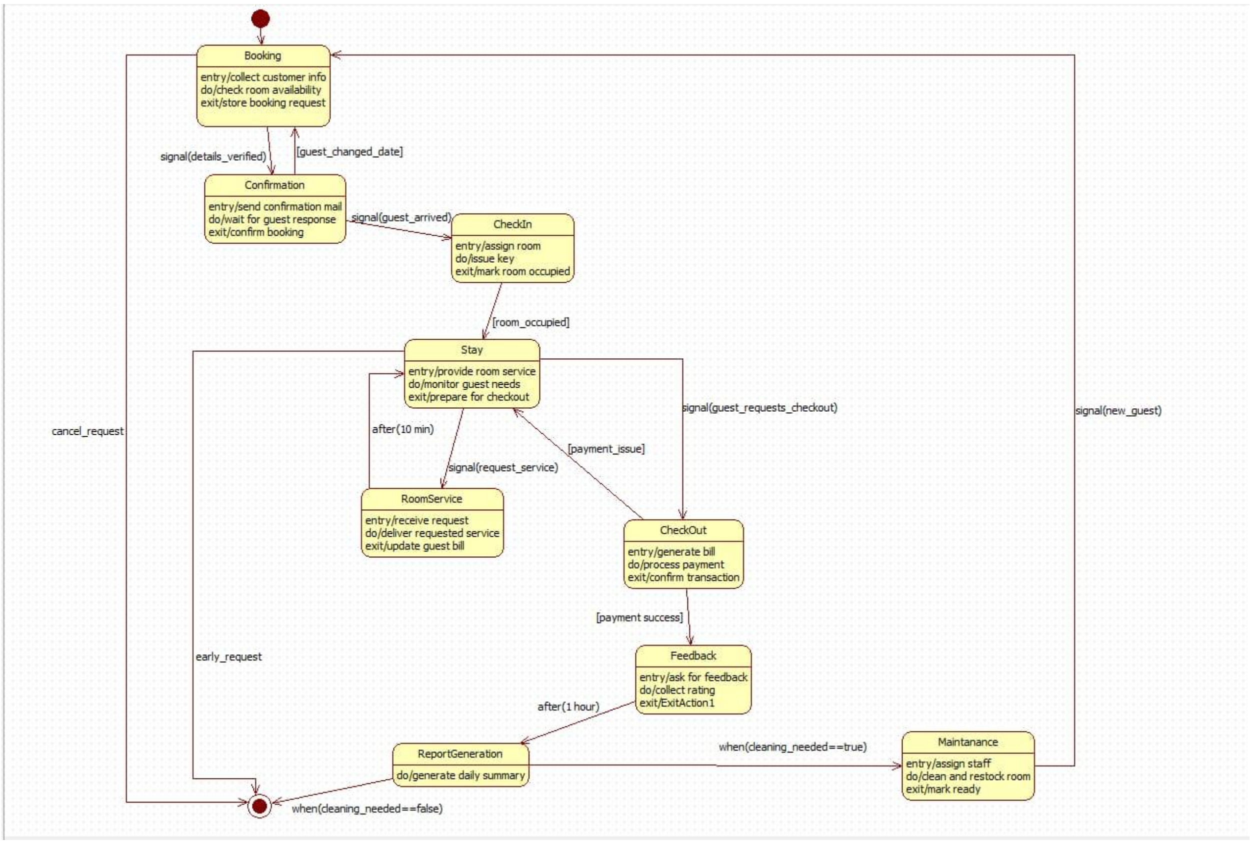


Figure 1.2: State Diagram for Hotel Management System

The hotel guest process begins with a booking stage where customer information is collected and room availability is checked, after which the request moves to confirmation, where the system sends a confirmation email and waits for the guest's response. Once the guest arrives, the process transitions to check-in, where a room is assigned and the key is issued before the guest enters the stay phase. During the stay, the hotel monitors guest needs and handles room-service requests, which temporarily move the workflow to a dedicated room-service state before returning to the stay state. When the guest requests checkout, the process moves to the checkout state, where the bill is generated and payment is processed; upon successful payment, the guest is directed to provide feedback. After checkout, if cleaning is required, the room enters a maintenance state where staff clean and restock the room before marking it ready; otherwise, the system proceeds to generate daily reports and waits for the next new guest to begin the cycle again.

Use Case Diagram

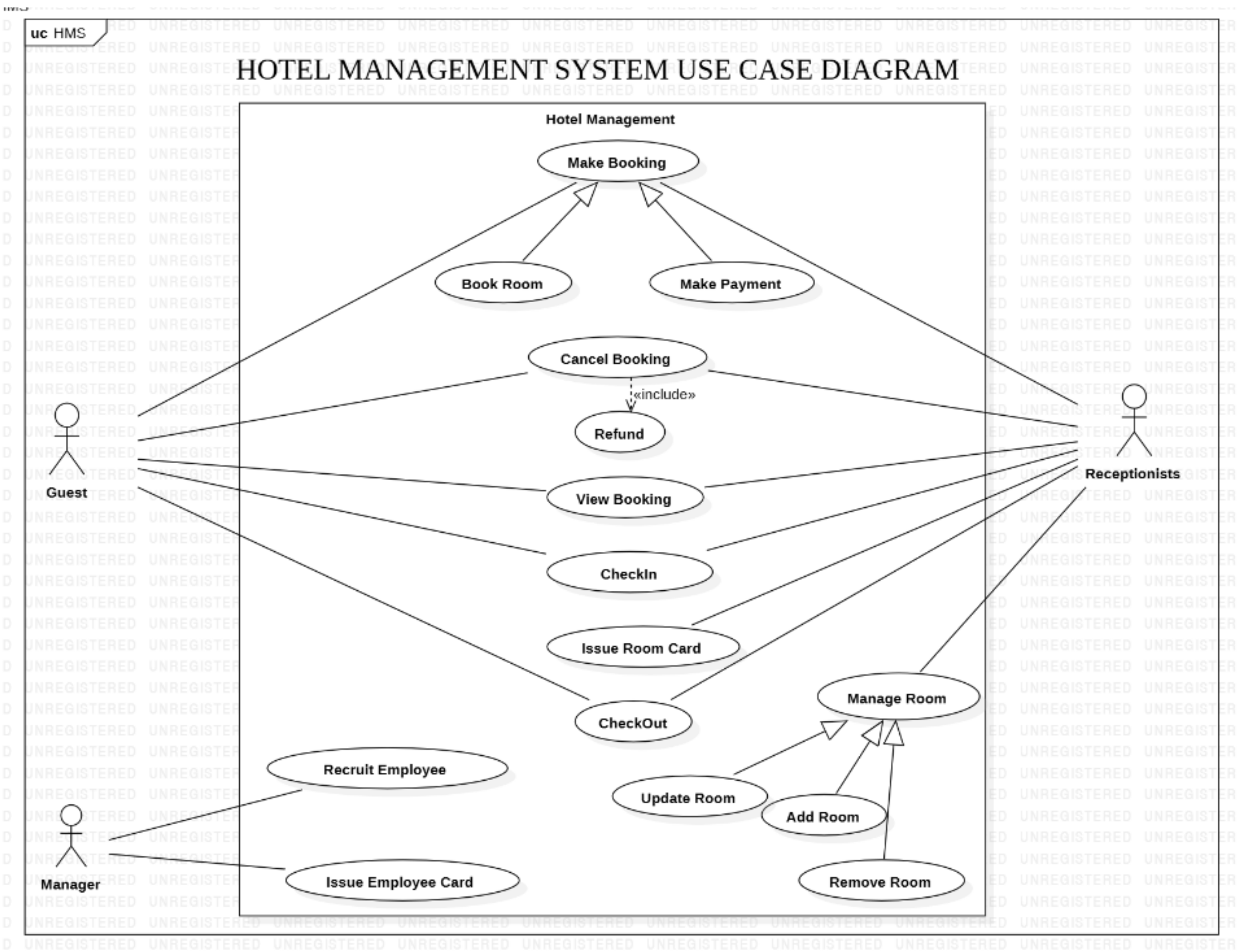


Figure 1.3: Use Case Diagram for Hotel Management System

The hotel management system allows guests to book rooms, make payments, cancel bookings, request refunds, view their bookings, check in, receive a room card, and check out. Receptionists support many of these operations by handling room bookings, payments, check-ins, issuing room cards, and managing room-related tasks such as updating, adding, or removing rooms. The manager, on the other hand, oversees staff-related functions including recruiting employees and issuing employee cards. The system organizes all these interactions under the broader hotel management process, ensuring that guests can access essential services while staff and managers maintain hotel operations smoothly.

Sequence Diagram

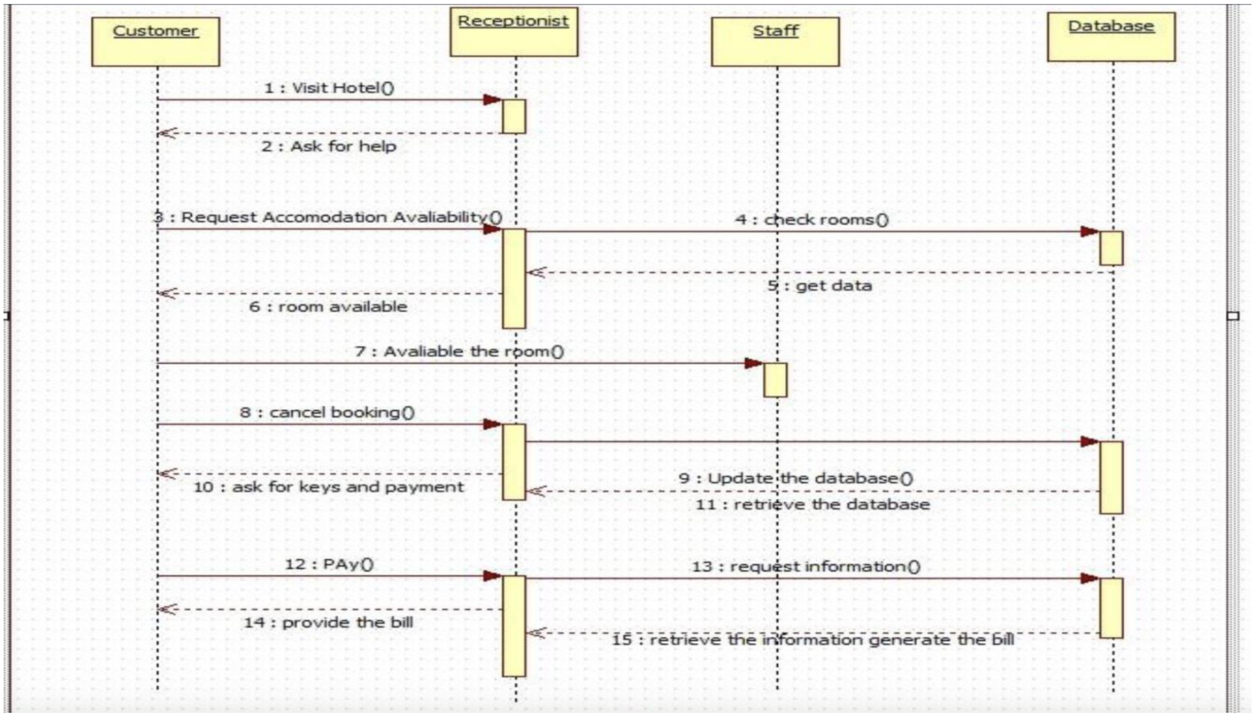

Customer
Receptionist
Staff
Database
1 : Visit Hotel()
2 : Ask for help
3 : Request Accomodation Avaliability()
4 : check rooms()
5 : get data
6 : room available
7 : Avaliable the room()
8 : cancel booking()
9 : Update the database()
10 : ask for keys and payment
11 : retrieve the database
12 : PAy()
13 : request information()
14 : provide the bill
15 : retrieve the information generate the bill

Figure 1.4: Sequence Diagram for Hotel Management System

In this hotel sequence, a customer first visits the hotel and asks the receptionist for assistance, then requests information about room availability. The receptionist contacts the staff, who check the rooms by querying the database and return the data, allowing the receptionist to inform the customer that a room is available. If the customer decides to cancel a booking, the receptionist updates the database accordingly. Later, when the customer asks for the room keys and proceeds to make payment, the receptionist requests billing information from the database, which retrieves the necessary details so the receptionist can provide the final bill to the customer.

Activity Diagram

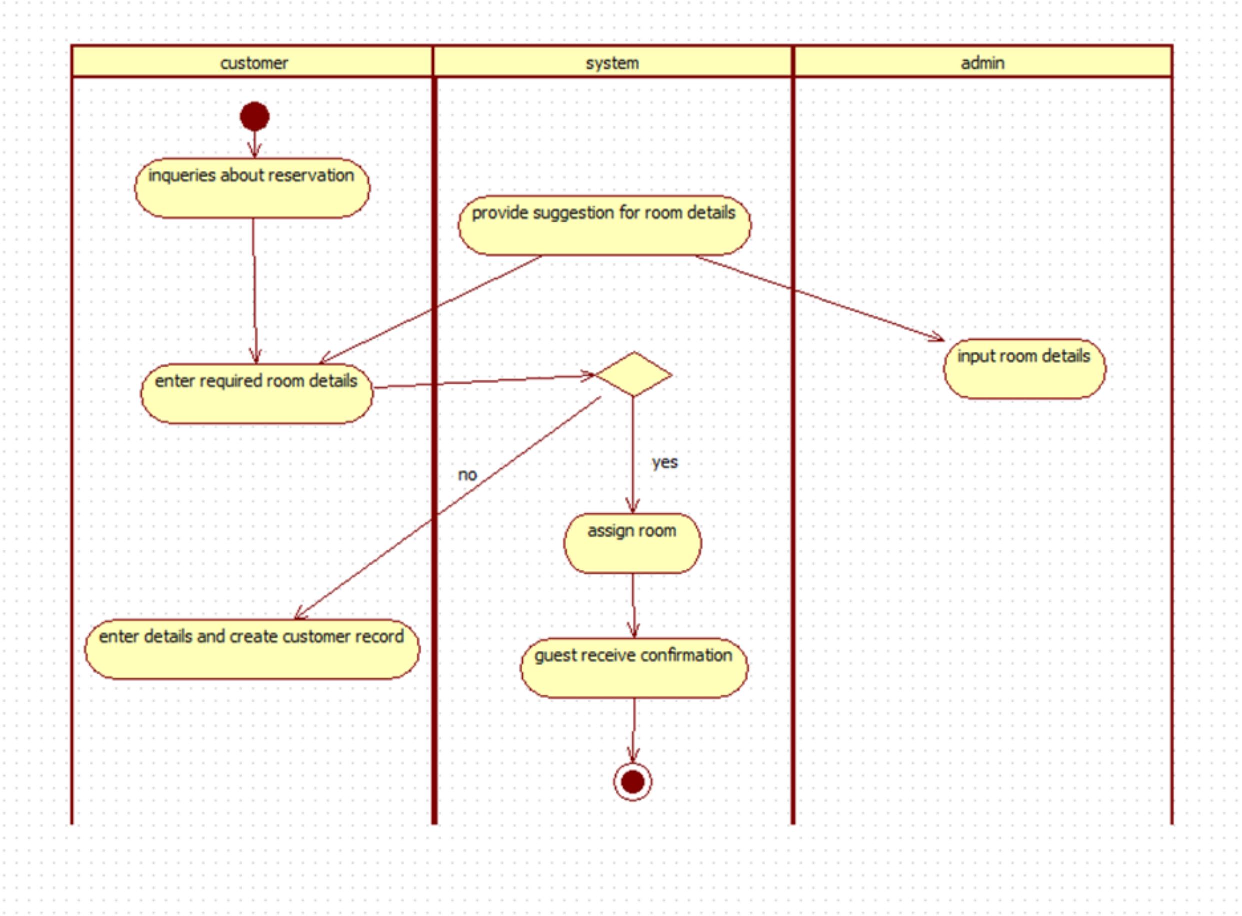

customer
system
admin
inqueries about reservation
provide suggestion for room details
input room details
enter required room details
no
yes
assign room
enter details and create customer record
guest receive confirmation

Figure 1.5: Activity Diagram for Hotel Management System

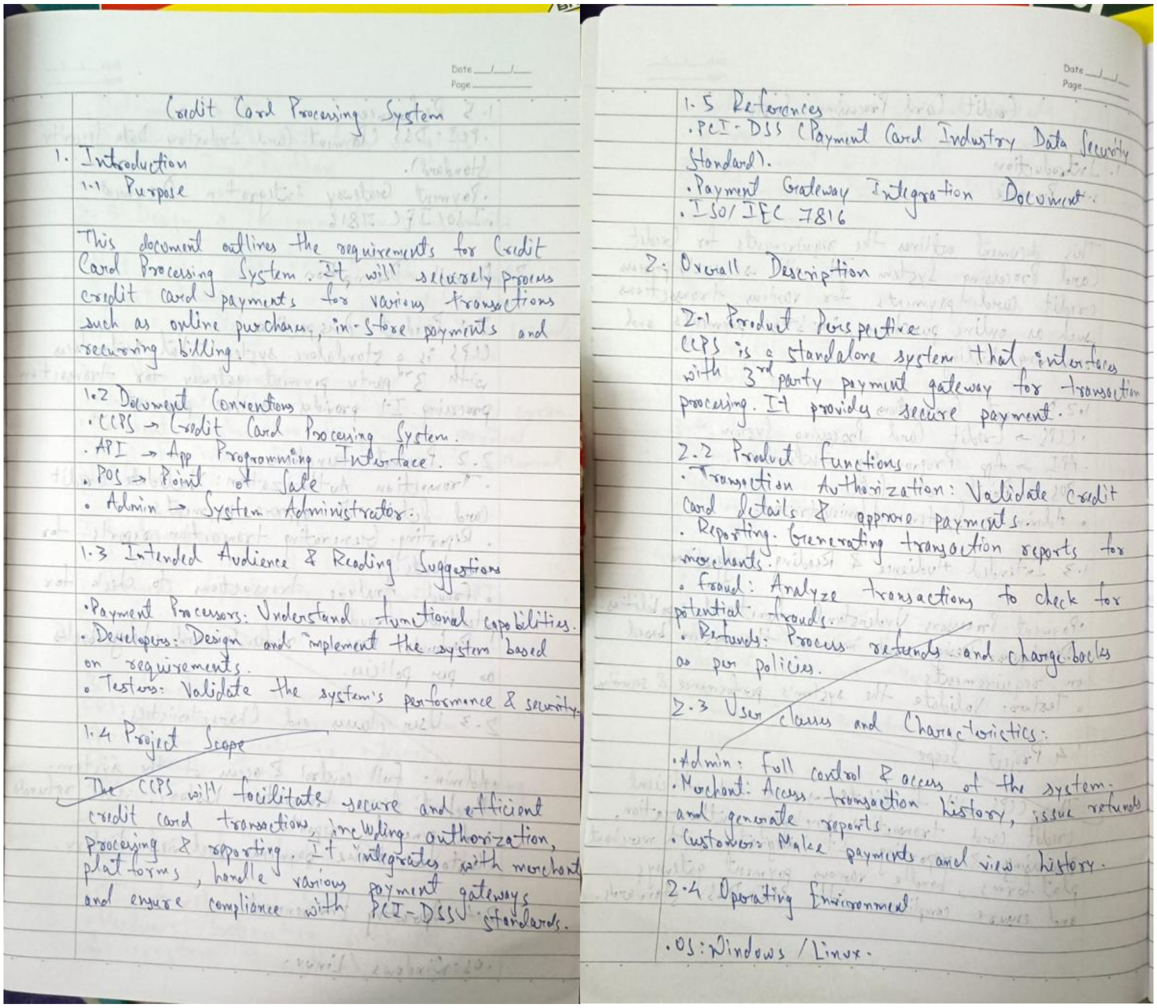
The activity begins when a customer inquires about making a reservation and proceeds to enter their required room details. The system then provides suggestions for suitable rooms while the admin inputs room information to help the system make a decision. Based on these details, the system checks if the room requirements can be met; if not, the customer is prompted to re-enter their details and a customer record is created. If the room is available, the system assigns the room and sends a confirmation to the guest, completing the reservation process.

2. Credit Card Processing

Problem Statement:

The Credit Card Processing System should allow cardholders to perform transactions such as purchases, cash withdrawals, balance inquiries, and bill payments. When a transaction is initiated, the system must verify card details, validate the cardholder's identity, check available credit limits, and ensure that the transaction is not fraudulent. The system must then approve or decline the transaction based on predefined rules and update the cardholder's account accordingly. Additionally, the system should generate periodic statements, track outstanding balances, calculate interest, and maintain accurate records of all transactions.

SRS-Software Requirements Specification



Date __/__/__
Page ____

Credit Card Processing System

1. Introduction

1.1 Purpose

This document outlines the requirements for Credit Card Processing System. It will securely process credit card payments for various transactions such as online purchases, in-store payments and recurring billing.

1.2 Document Conventions

- CCPS → Credit Card Processing System.
- API → App Programming Interface.
- POS → Point of Sale
- Admin → System Administrator.

1.3 Intended Audience & Reading Suggestions

- Payment Processors: Understand functional capabilities.
- Developers: Design and implement the system based on requirements.
- Testers: Validate the system's performance & security.

1.4 Project Scope

The CCPS will facilitate secure and efficient credit card transactions, including authorization, processing & reporting. It integrates with merchant platforms, handle various payment gateways and ensure compliance with PCI-DSS standards.

Date __/__/__
Page ____

1.5 References

- PCI-DSS (Payment Card Industry Data Security Standard).
- Payment Gateway Integration Document.
- ISO/IEC 7816

2. Overall Description

2.1 Product Perspective

CCPS is a standalone system that interfaces with 3rd party payment gateway for transaction processing. It provides secure payment.

2.2 Product functions

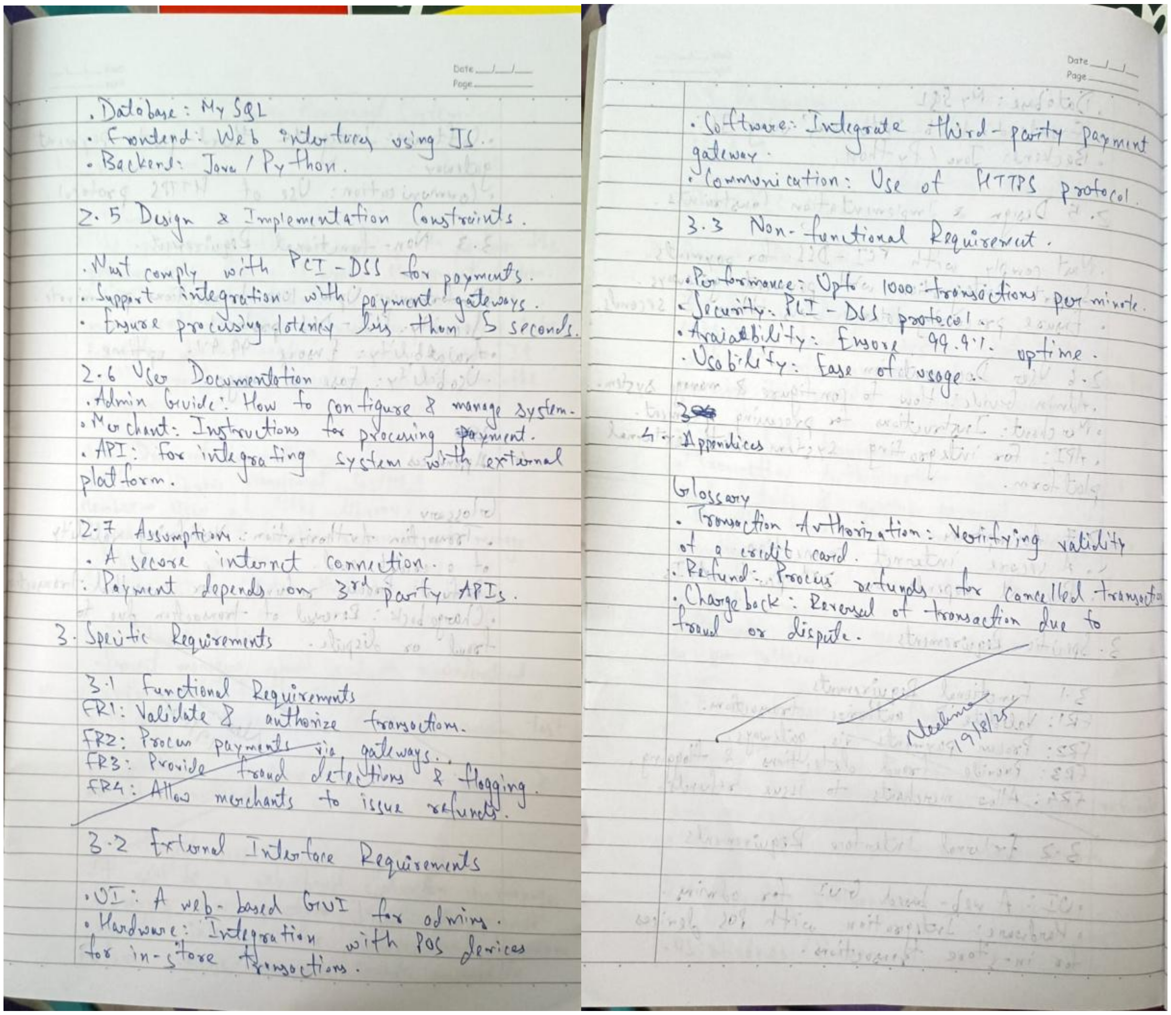
- Transaction Authorization: Validate credit card details & approve payments.
- Reporting: Generating transaction reports for merchants.
- Fraud: Analyze transactions to check for potential frauds.
- Refunds: Process refunds and charge backs as per policies.

2.3 User classes and Characteristics:

- Admin: Full control & access of the system.
- Merchant: Access transaction history, issue refunds and generate reports.
- Customer: Make payments and view history.

2.4 Operating Environment

- OS: Windows / Linux.



Date __/__/__
Page ____

. Datobase : My SQL
· Frontend: Web intertuca using JS.
· Backend: Java / Python.

2. 5 Design & Implementation Constraints.

. Must comply with PCI - DSS for payments.
· Support integration with payment gateways.
· Ensure processing latency less than 5 seconds.

2. 6 User Documentation
. Admin Guide: How to configure & manage system.
. Merchant: Instructions for processing payment.
. API: for integrating system with external platform.

2. 7 Assumptions:
· A secore internet connection.
: Payment depends on 3rd party APIs.

3. Specific Requirements

3.1 Functional Requirements
FR1: Validate & authorize transactiom.
FR2: Process payments via gateways.
FR3: Provide fraud detections & flagging.
FR4: Allow merchants to issue refunds.

3.2 External Interface Requirements

· UI: A web-based GUI for admins.
· Hardware: Integration with POS devices for in-store transactions.

Date __/__/__
Page ____

· Software: Integrate third-party payment gateway.
· Communication: Use of HTTPS protocol.

3.3 Non-functional Requirement.

· Performance: Upto 1000 transactions per minute.
· Security: PCI - DSS protocol.
· Avaiabbility: Ensure 99.9% uptime.
· Usability: Ease of usage.

~~3~~
4. Appendices

Glossary
· Transaction Authorization: Verifying validity of a credit card.
· Refund: Process refunds for cancelled transaction
· Chargeback: Reversal of transaction due to fraud or dispute.

Neelima 19/8/25

Class Diagram

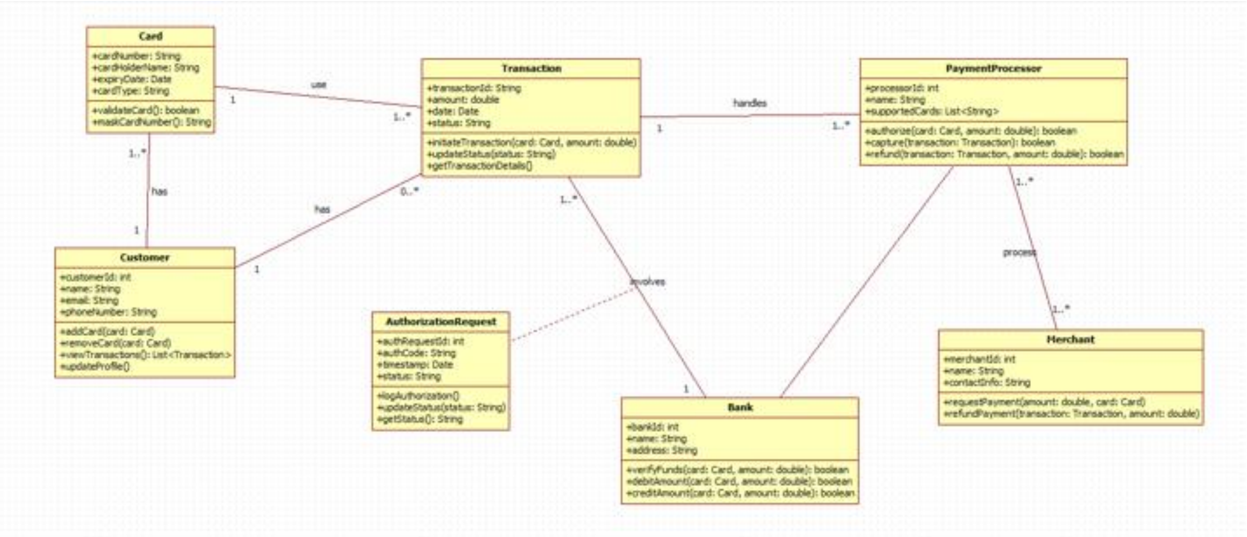


Figure 2.1: Class Diagram for Credit Card Processing

This payment processing system manages how customers use their cards to make transactions with merchants. Each customer owns one or more cards, and when a transaction is initiated, the card information is used to create a transaction request. The payment processor handles these transactions by forwarding authorization requests to the bank, which verifies card details, checks account status, and approves or declines the request. Once authorized, the transaction is processed for the merchant, who can also request refunds through the system. Authorization requests record details such as status and time, while the bank maintains operations for authorization, refund, and transaction validation. Throughout the process, all components—customer, card, payment processor, bank, and merchant—interact to ensure secure and accurate handling of payments.

State Diagram

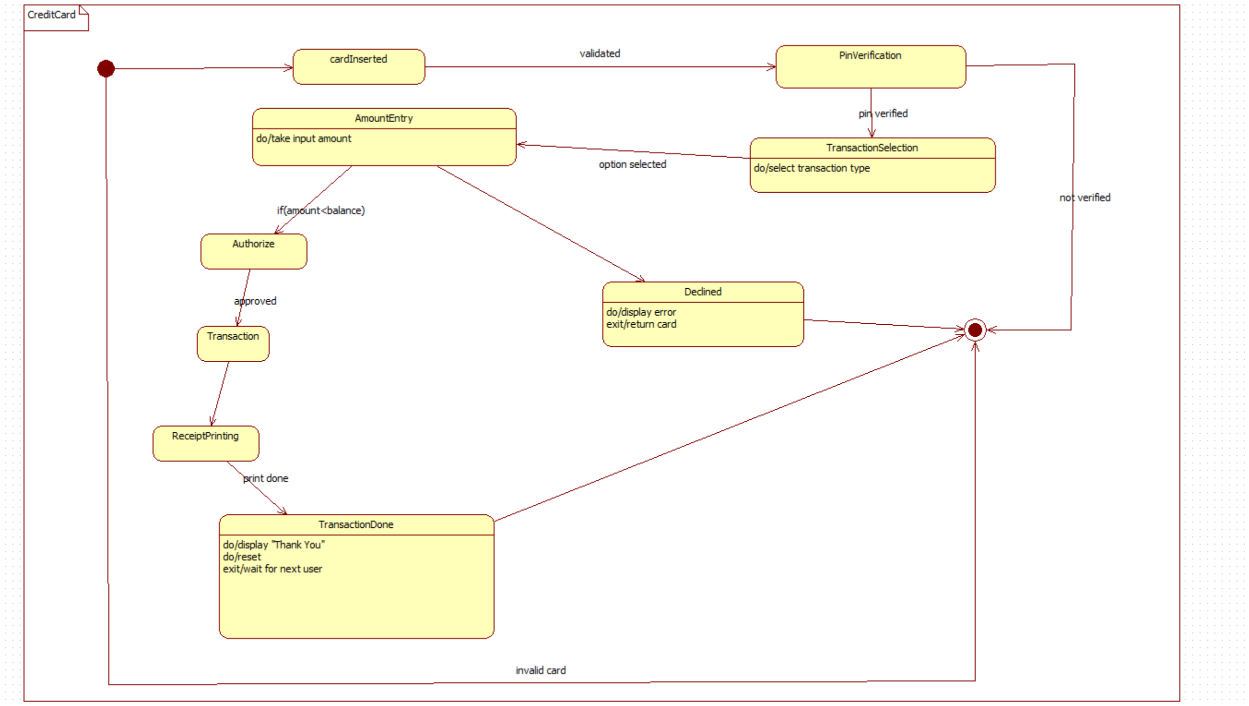


Figure 2.2: State Diagram for Credit Card Processing

The credit card transaction process begins when a user inserts their card, after which the system validates it and prompts for PIN verification. If the PIN is correct, the user proceeds to select the type of transaction and then enters the required amount. The system checks whether the entered amount is within the available balance; if it is, the transaction is authorized, executed, and followed by receipt printing. Once the receipt is generated, the system displays a thank-you message, resets itself, and waits for the next user. If the card is invalid, the PIN is incorrect, or the amount exceeds the balance, the process moves to a declined state where an error is displayed and the card is returned, ending the workflow.

Use Case Diagram

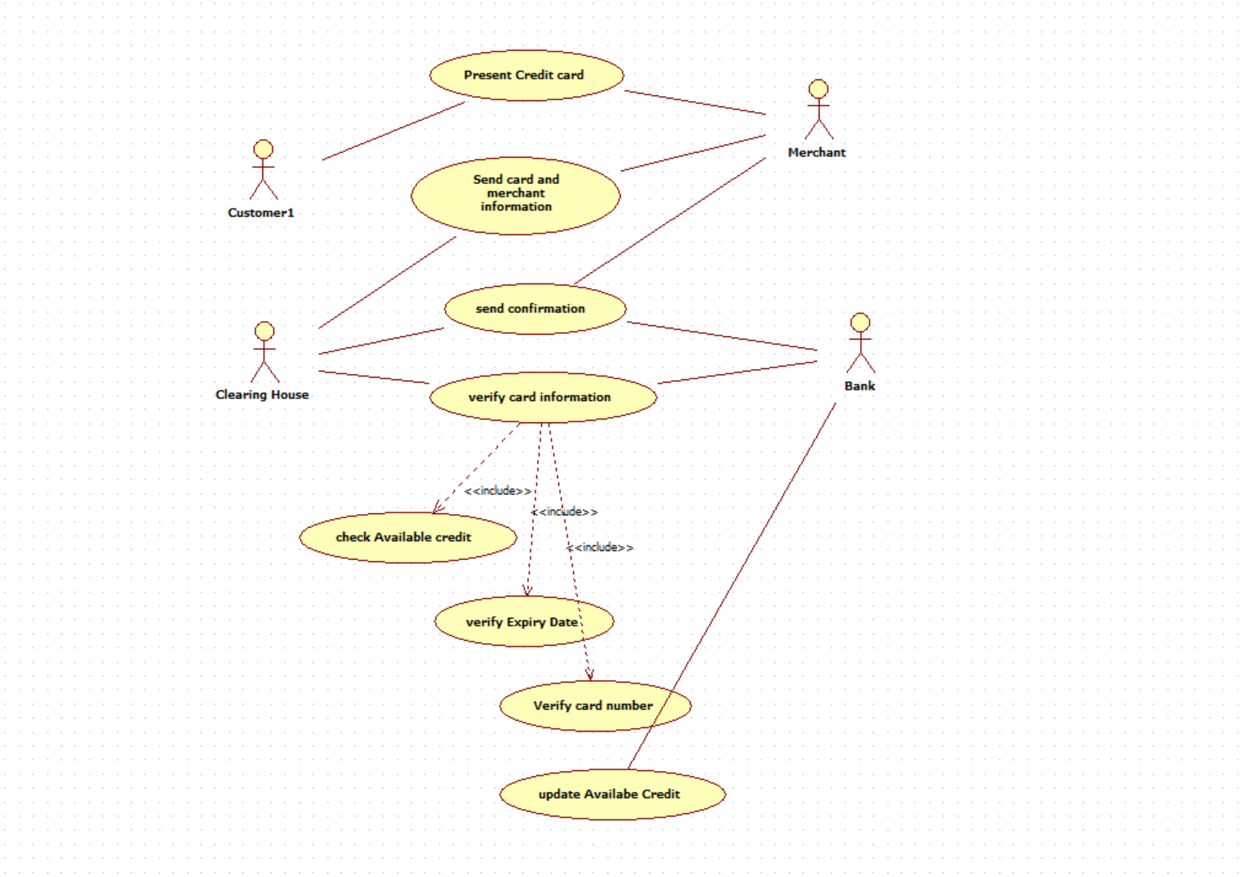


Figure 2.3: Use Case Diagram for Credit Card Processing

In this credit card authorization process, a customer presents their credit card to the merchant, who sends the card and transaction details to the clearing house for verification. The clearing house then communicates with the bank to verify the card by performing several checks, including validating the card number, confirming the expiry date, and checking whether sufficient available credit remains. Once these checks are completed, the clearing house sends a confirmation back to the merchant about whether the transaction is approved, and the bank updates the available credit accordingly. This cooperation among customer, merchant, clearing house, and bank ensures that every card transaction is securely authenticated before completion.

Sequence Diagram

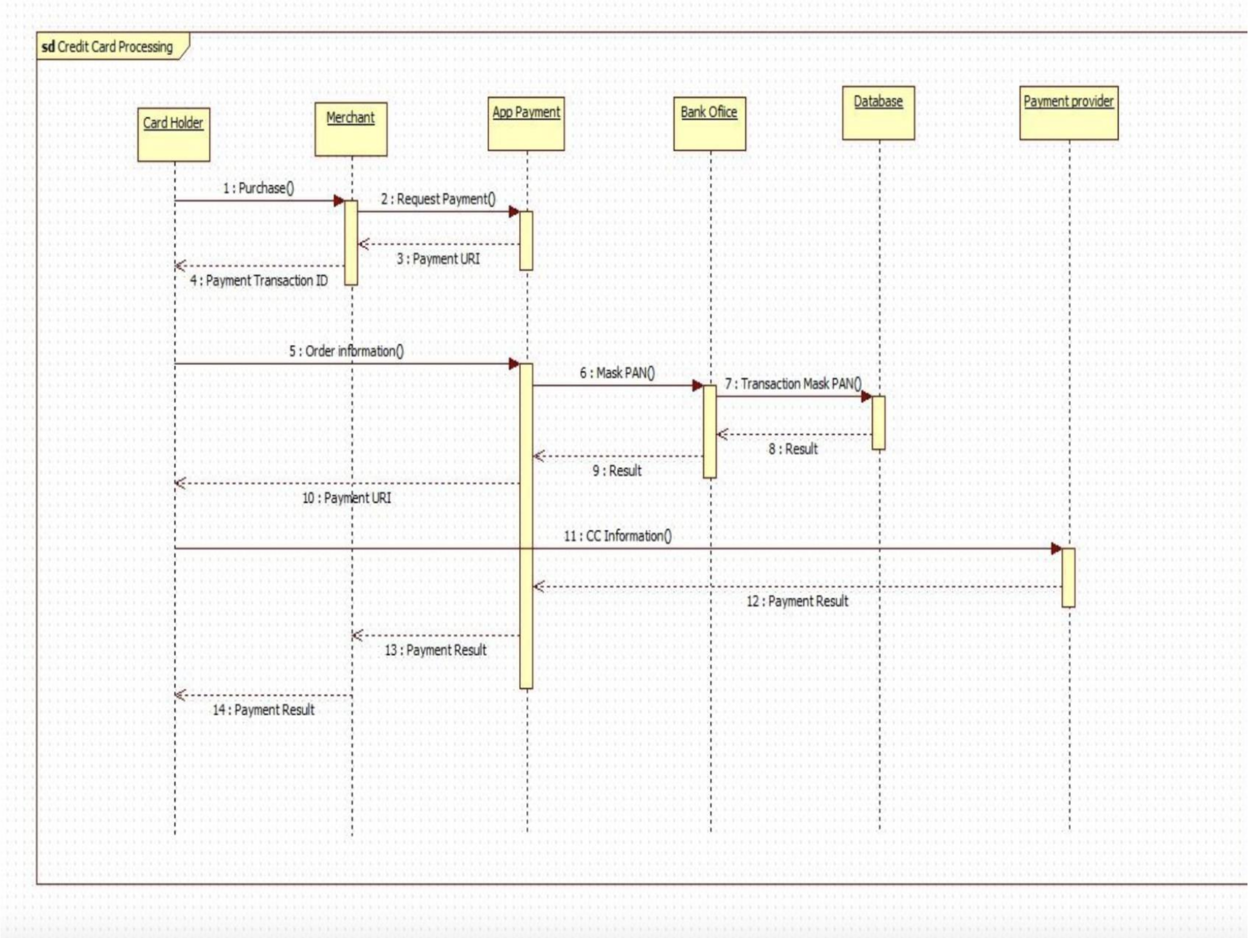


Figure 2.4: Sequence Diagram for Credit Card Processing

In this credit card processing sequence, the card holder begins a purchase request, which the merchant forwards to the app payment system to obtain a payment URL. After receiving the payment transaction ID and payment URL, the merchant sends the customer’s order information to the bank office, which masks the card number (PAN) and forwards the masked transaction details to the database for verification. The database returns the result to the bank office, which passes it back through the app payment system along with the credit card information. The app payment system processes the details with the payment provider, receives the final payment result, and returns it to the merchant, who ultimately notifies the customer of the completed transaction.

Activity Diagram

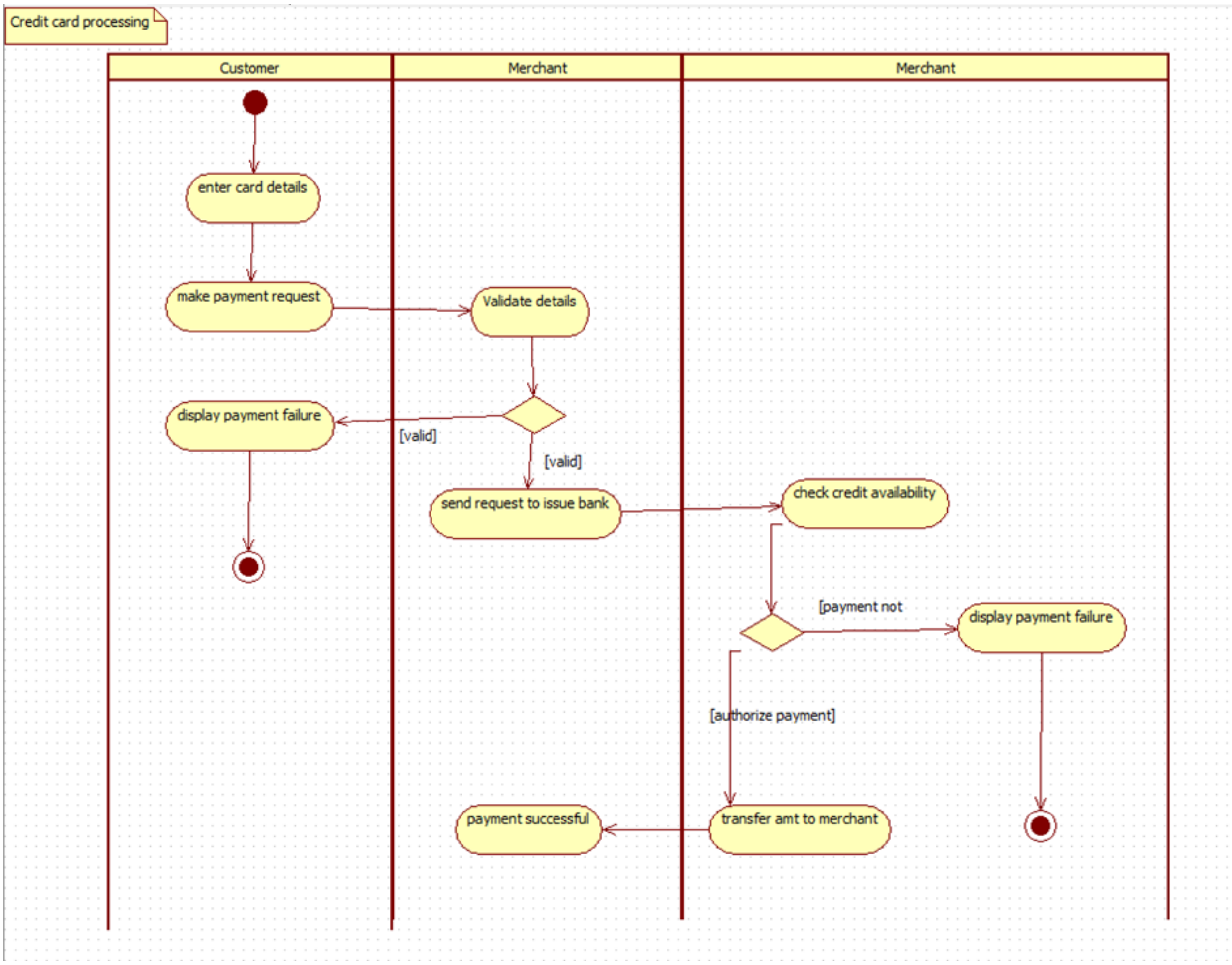


Figure 2.5: Activity Diagram for Credit Card Processing

The credit card payment process begins when the customer enters their card details and submits a payment request to the merchant. The merchant first validates the provided information; if the details are invalid, the system immediately displays a payment failure message to the customer. If the details are correct, the merchant forwards the request to the issuing bank, where the system checks the availability of credit. If sufficient credit is not available, the bank notifies the merchant, who then informs the customer of the payment failure. When credit is available and the payment is authorized, the bank transfers the approved amount to the merchant, and the system confirms that the payment was successful, completing the process.

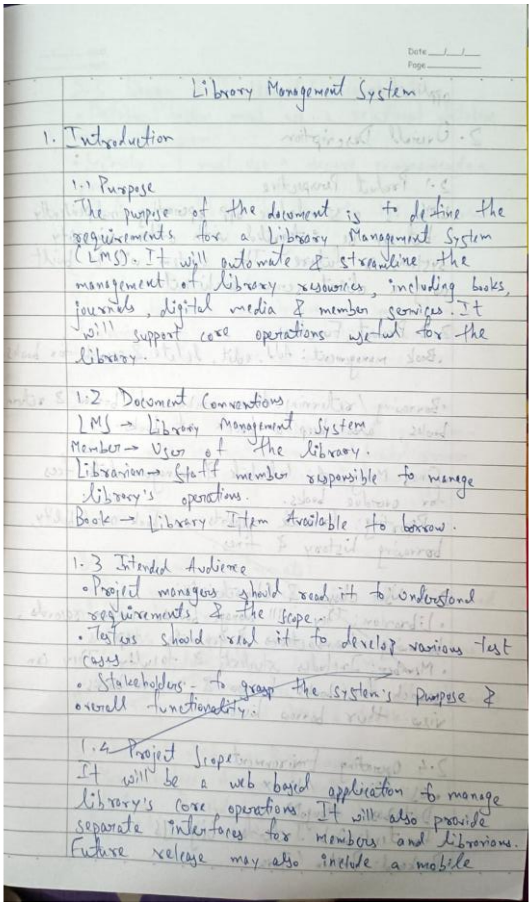
3. Library Management System

Problem Statement:

Libraries today manage large volumes of books, journals, digital media, and member records. Many libraries still rely on manual methods such as record books, paper-based membership files, and handwritten issue–return logs. These manual processes are time-consuming, prone to human errors, difficult to update, and inefficient when handling large numbers of users and resources. To overcome these limitations and modernize library operations, an automated Library Management System is required.

The Library Management System should enable librarians to efficiently manage the acquisition, cataloging, tracking, and distribution of books and other resources. The system must allow members to search for books, check availability, borrow and return items, place reservations, and receive notifications for due dates or overdue fines. Librarians should be able to register new members, maintain member details, add or remove books, update catalog information, manage book categories, handle book issues, enforce return deadlines, calculate fines for late returns, and generate various reports related to inventory and member activity.

SRS-Software Requirements Specification



Date __/__/__
Page ______

Library Management System

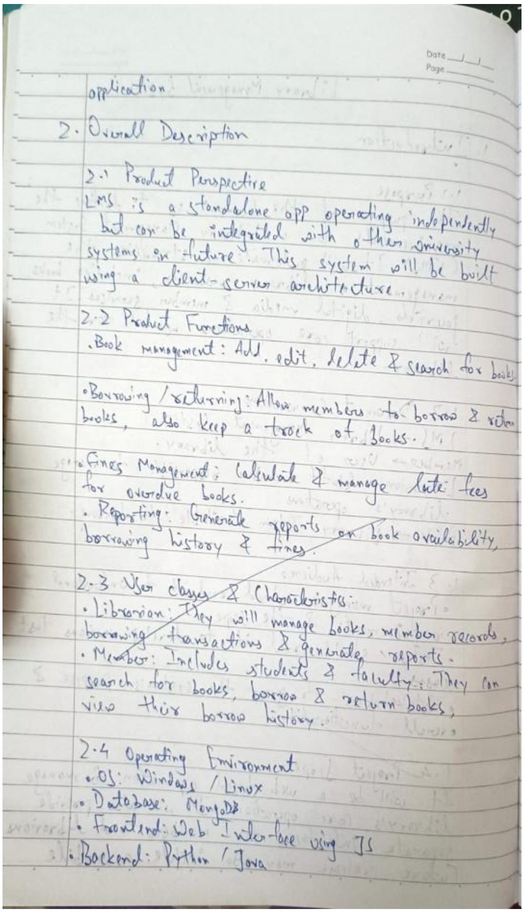
1. Introduction

1.1 Purpose
The purpose of the document is to define the requirements for a Library Management System (LMS). It will automate & streamline the management of library resources, including books, journals, digital media & member services. It will support core operations useful for the library.

1.2 Document Conventions
LMS → Library Management System
Member → User of the library.
Librarian → Staff member responsible to manage library's operations.
Book → Library Item Available to borrow.

1.3 Intended Audience
- Project managers should read it to understand requirements & the scope.
- Testers should read it to develop various test cases.
- Stakeholders - to grasp the system's purpose & overall functionality.

1.4 Project Scope
It will be a web based application to manage library's core operations. It will also provide separate interfaces for members and librarians. Future release may also include a mobile



Date __/__/__
Page ______

application

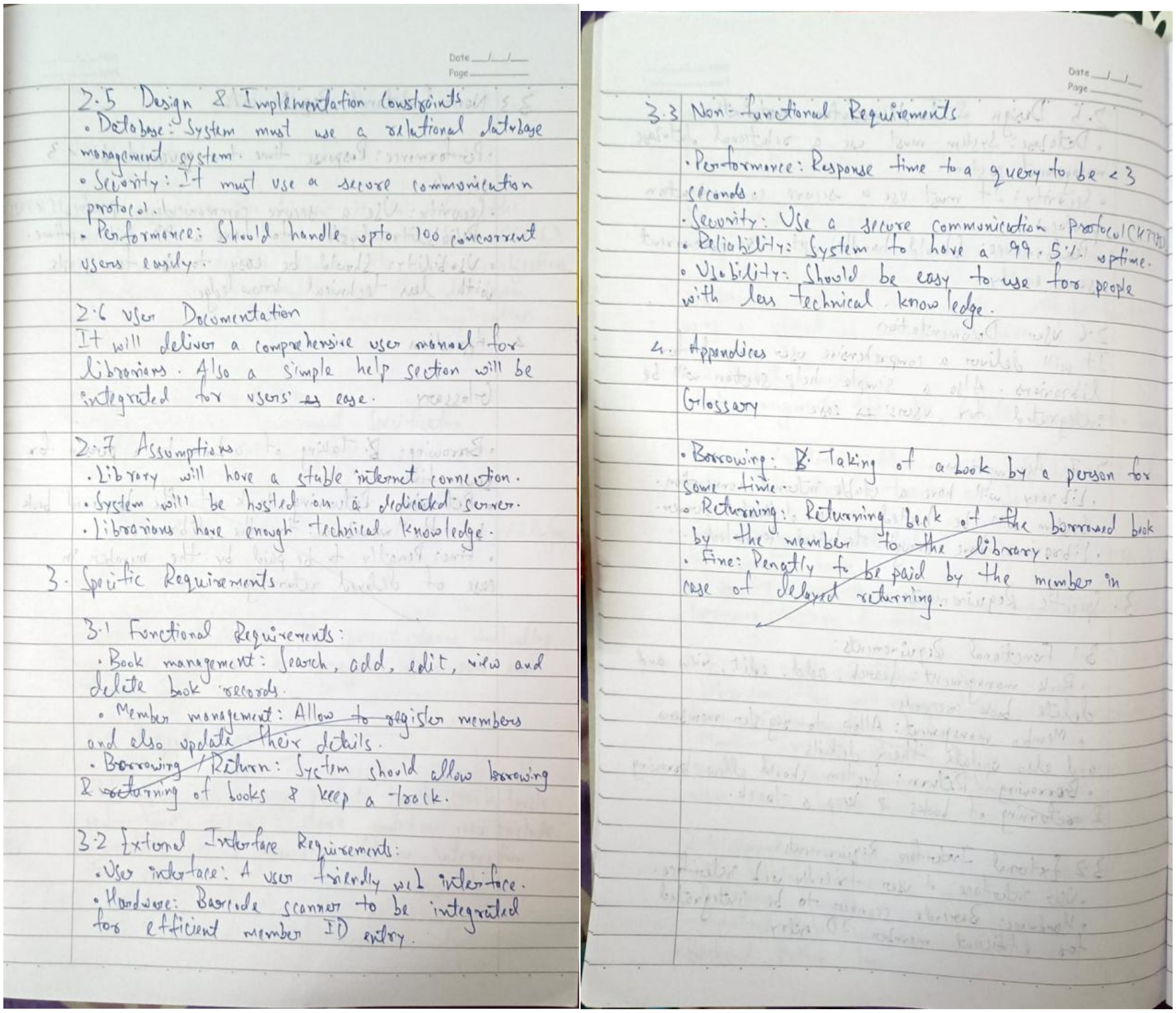
2. Overall Description

2.1 Product Perspective
LMS is a standalone app operating independently but can be integrated with other university systems in future. This system will be built using a client-server architecture.

2.2 Product Functions
- Book management: Add, edit, delete & search for books
- Borrowing / returning: Allow members to borrow & return books, also keep a track of books.
- Fines Management: Calculate & manage late fees for overdue books.
- Reporting: Generate reports on book availability, borrowing history & fines.

2.3 User classes & Characteristics:
- Librarian: They will manage books, member records, borrowing transactions & generate reports.
- Member: Includes students & faculty. They can search for books, borrow & return books, view their borrow history.

2.4 Operating Environment
- OS: Windows / Linux
- Database: MongoDB
- Frontend: Web Interface using JS
- Backend: Python / Java



Date __/__/__
Page ______

2.5 Design & Implementation Constraints
- Database: System must use a relational database management system.
- Security: It must use a secure communication protocol.
- Performance: Should handle upto 100 concurrent users easily.

2.6 User Documentation
It will deliver a comprehensive user manual for librarians. Also a simple help section will be integrated for users' ~~as~~ ease.

2.7 Assumptions
- Library will have a stable internet connection.
- System will be hosted on a dedicated server.
- Librarians have enough technical knowledge.

3. Specific Requirements

3.1 Functional Requirements:
- Book management: Search, add, edit, view and delete book records.
- Member management: Allow to register members and also update their details.
- Borrowing / Return: System should allow borrowing & returning of books & keep a track.

3.2 External Interface Requirements:
- User interface: A user friendly web interface.
- Hardware: Barcode scanner to be integrated for efficient member ID entry.

Date __/__/__
Page ______

3.3 Non-functional Requirements
- Performance: Response time to a query to be < 3 seconds.
- Security: Use a secure communication protocol (HTTPS)
- Reliability: System to have a 99.5% uptime.
- Usability: Should be easy to use for people with less technical knowledge.

4. Appendices

Glossary

- Borrowing: ~~B.~~ Taking of a book by a person for some time.
- Returning: Returning back of the borrowed book by the member to the library.
- Fine: Penatly to be paid by the member in case of delayed returning.

Class Diagram

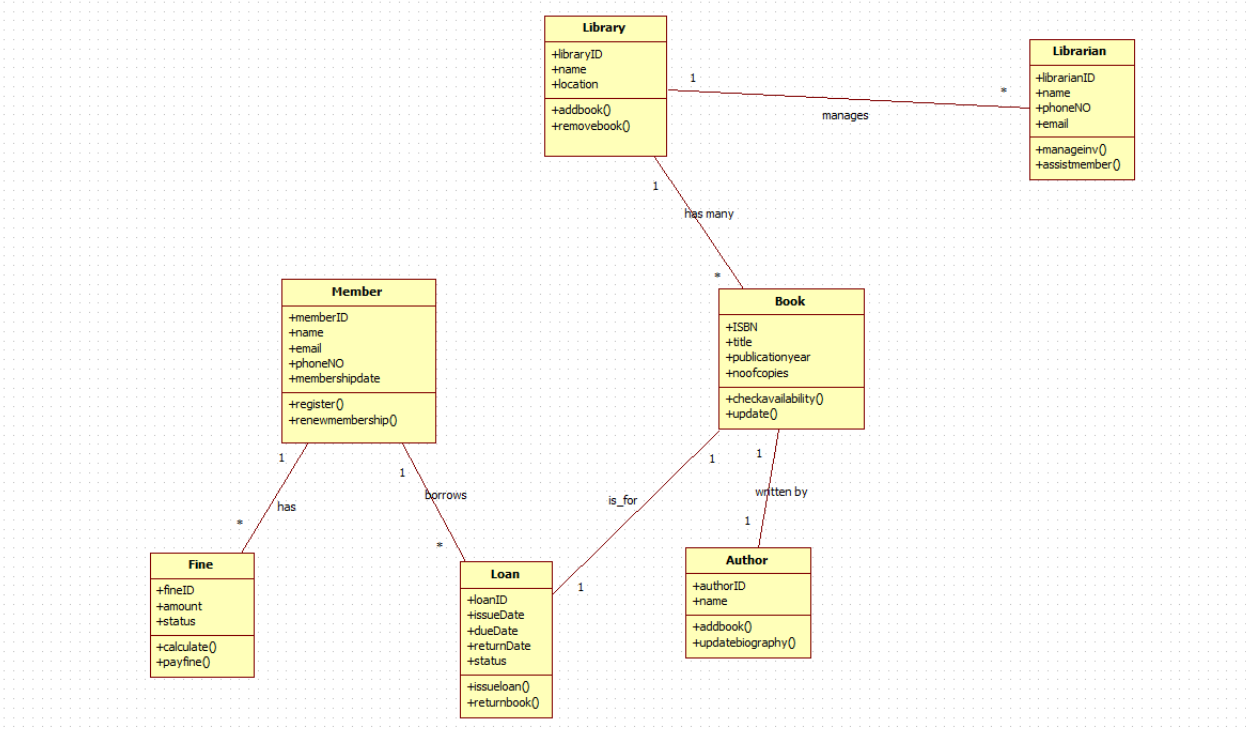


Figure 3.1: Class Diagram for Library Management System

In this library management system, a library contains many books, each written by one or more authors and managed by librarians who oversee book records and assist members. Members can register with the library and borrow books through loan records that track issue dates, due dates, and return dates. If a member returns a book late or violates library rules, fines are generated and linked to the member. The system ensures that book availability is updated, loans are properly recorded, fines are calculated, and librarians can manage inventory and support member needs, while authors and their books are also maintained for accurate cataloging.

State Diagram

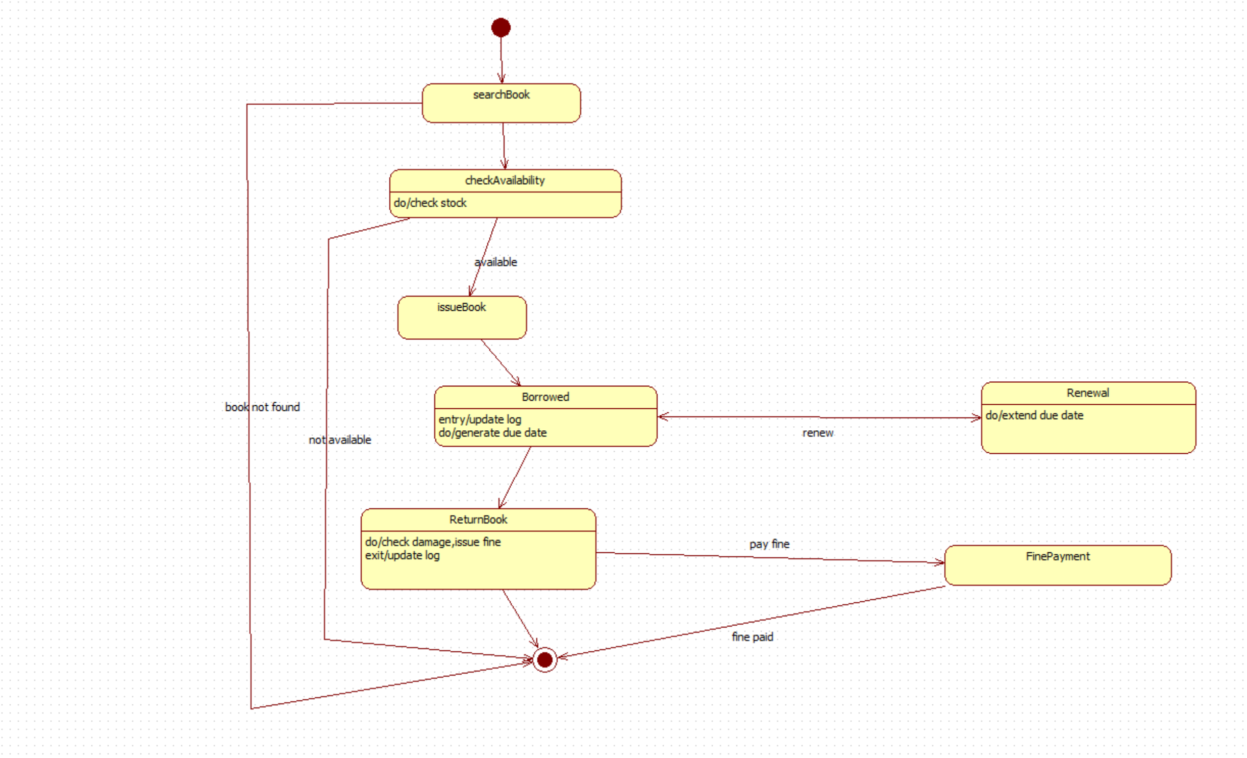


Figure 3.2: State Diagram for Library Management System

The book-borrowing process starts when a user searches for a book, after which the system checks its availability in the library's stock. If the book is found and available, it is issued to the user, who enters the borrowed state where the system logs the transaction and generates a due date. From this state, the user may choose to renew the book, extending the due date, or return it. Upon return, the system checks for any damage or late submission and issues a fine if necessary; the user can then proceed to pay the fine, after which the record is updated and the process ends. If the book is not found or unavailable at the start, the system immediately terminates the process with an appropriate message.

Use Case Diagram

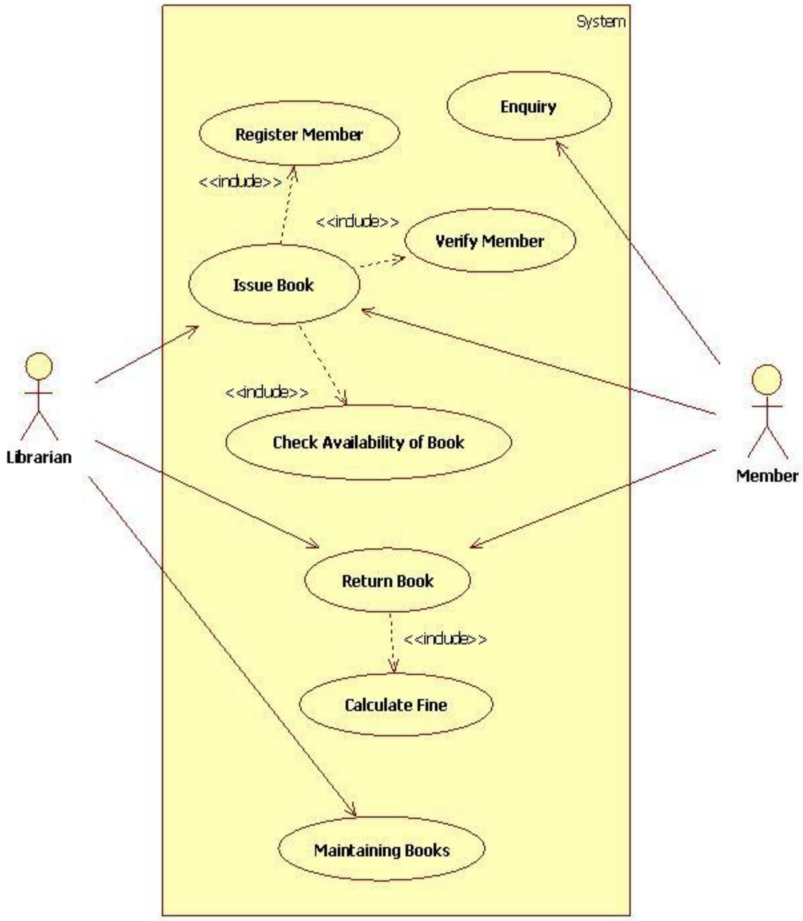


Figure 3.3: Use Case Diagram for Library Management System

In this library management system, members can make enquiries about books and have their membership verified before borrowing any items. The librarian is responsible for registering new members, issuing books, checking the availability of requested books, and processing book returns. When a book is returned, the system calculates any applicable fines for late or damaged returns. The librarian also oversees maintaining the collection, ensuring that books are properly managed and updated in the system. Each major task, such as issuing books or returning them, includes necessary supporting actions like verifying membership or checking book availability to maintain smooth and accurate library operations.

Sequence Diagram

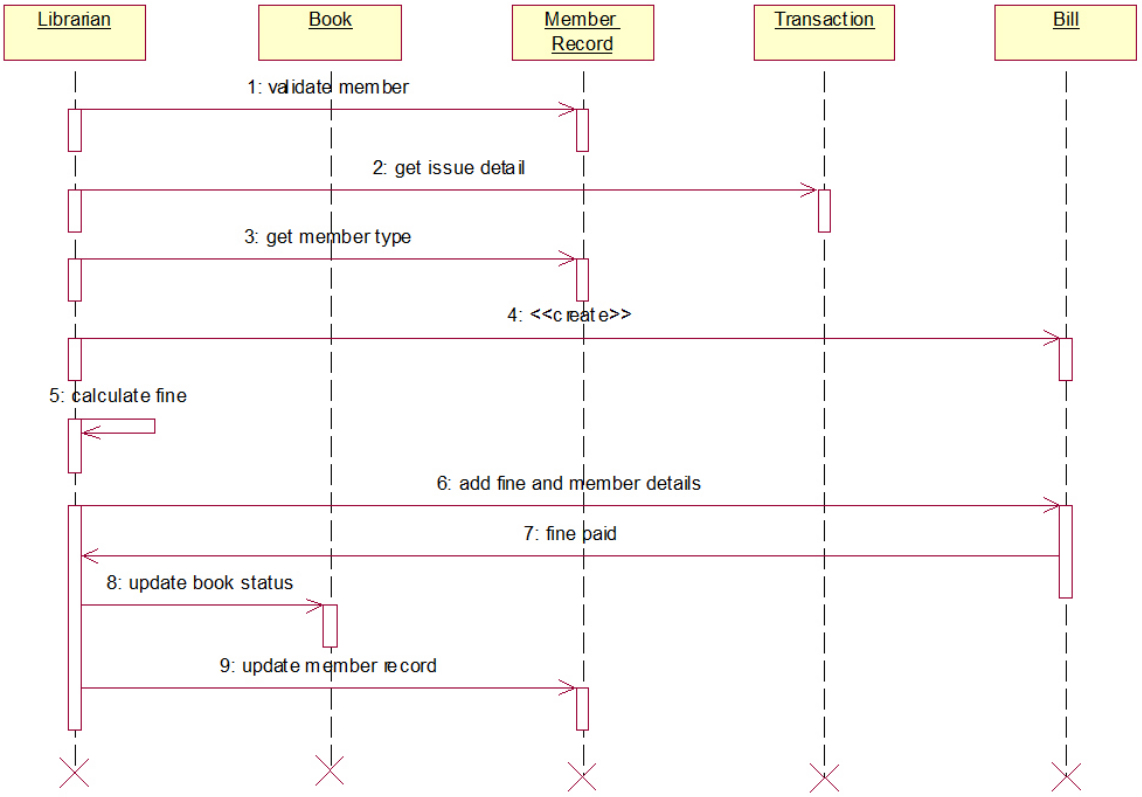


Figure 3.4: Sequence Diagram for Library Management System

In this fine-processing sequence, the librarian begins by validating the member and then requests issue details from the member record, followed by retrieving the member type to determine applicable rules. A fine calculation process is initiated, and the system creates a transaction entry before adding the fine and member information to the transaction. Once the member pays the fine, the system updates the status of the book and then updates the member record to reflect the completed transaction, concluding the process.

Activity Diagram

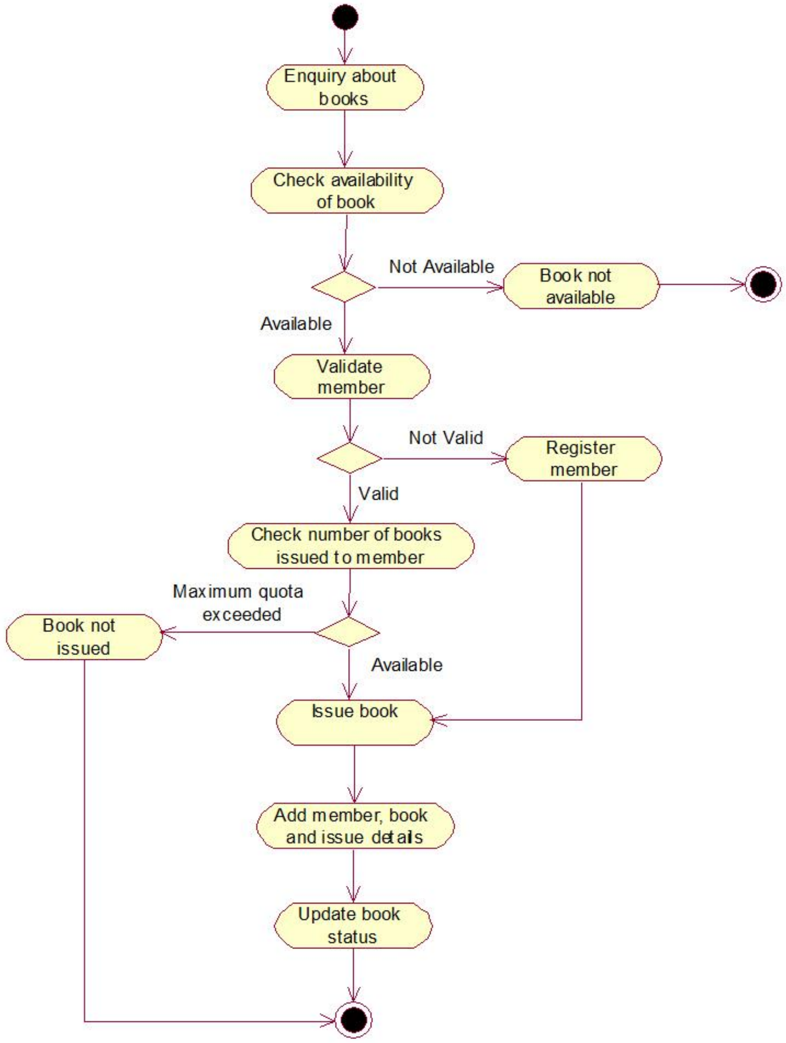


Figure 3.5: Activity Diagram for Library Management System

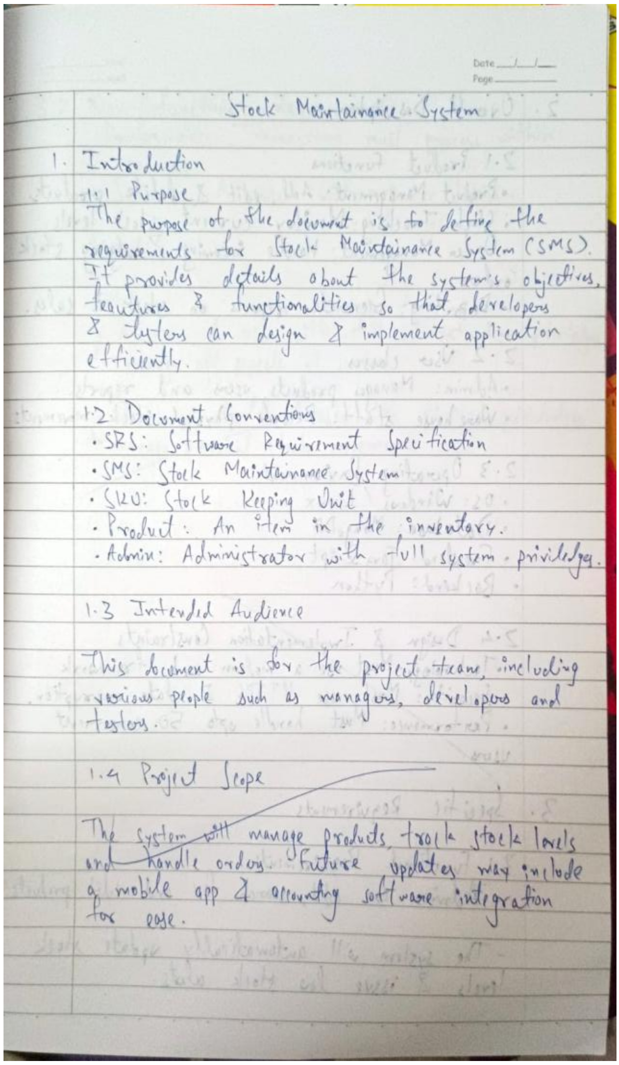
The library book-issuing process begins when a user makes an enquiry about a book, prompting the system to check its availability. If the book is not available, the process ends immediately. If available, the system proceeds to validate the member; an invalid member is redirected to the registration process before continuing. Once validated, the system checks how many books the member has already borrowed; if the borrowing limit is exceeded, the book cannot be issued. If the member is eligible, the book is issued, and the system records the member, book, and issue details before updating the book's status, successfully completing the transaction.

4. Stock Maintenance System

Problem Statement

The Stock Maintenance System should enable businesses to track stock quantities, monitor product movements, and maintain updated records of items entering or leaving the inventory. The system must allow users to add new stock items, categorize products, update quantities, record purchases, process stock issues, and generate alerts when stock levels fall below a specified threshold. It should also support search and inquiry functions so that users can quickly check availability, item details, supplier information, and stock status in real time. The system must maintain historical records of stock transactions, including purchases, sales, returns, damaged goods, and adjustments, ensuring transparency and traceability.

SRS-Software Requirements Specification



Date __/__/__
Page ____

Stock Maintainance System

1. Introduction

1.1 Purpose

The purpose of the document is to define the requirements for Stock Maintainance System (SMS). It provides details about the system's objectives, features & functionalities so that developers & testers can design & implement application efficiently.

1.2 Document Conventions

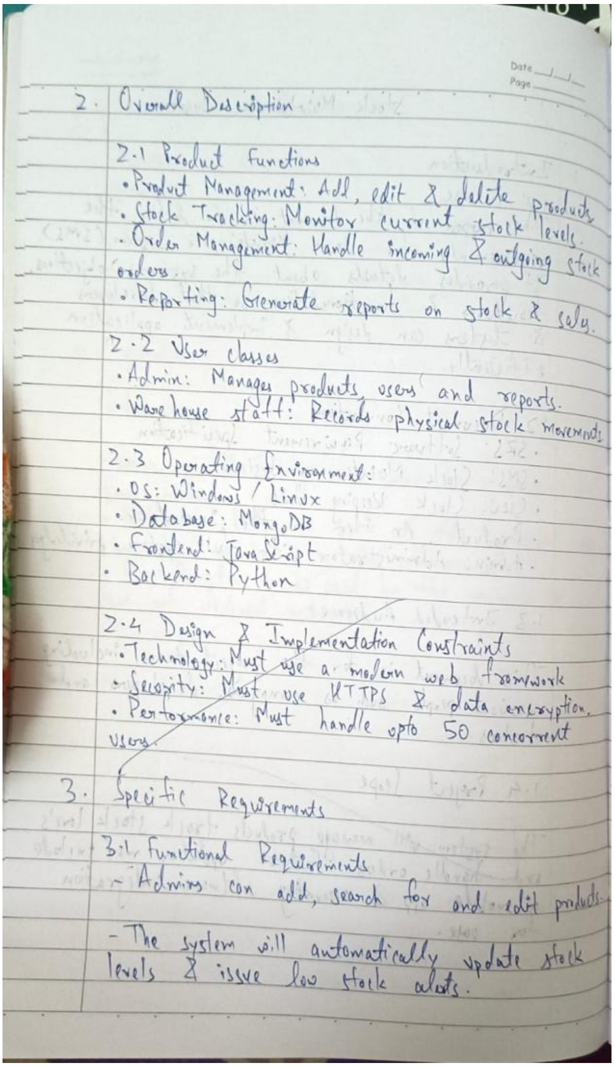
- SRS: Software Requirment Specification
- SMS: Stock Maintainance System
- SKU: Stock Keeping Unit
- Product: An item in the inventory.
- Admin: Administrator with full system privilidges.

1.3 Intended Audience

This document is for the project team, including various people such as managers, developers and testers.

1.4 Project Scope

The system will manage products, track stock levels and handle orders. Future updates may include a mobile app & accounting software integration for ease.



Date __/__/__
Page ____

2. Overall Description

2.1 Product Functions

- Product Management: Add, edit & delete products
- Stock Tracking: Monitor current stock levels.
- Order Management: Handle incoming & outgoing stock orders.
- Reporting: Generate reports on stock & sales.

2.2 User classes

- Admin: Manages products, users and reports.
- Warehouse staff: Records physical stock movements.

2.3 Operating Environment:

- OS: Windows / Linux
- Database: MongoDB
- Frontend: JavaScript
- Backend: Python

2.4 Design & Implementation Constraints

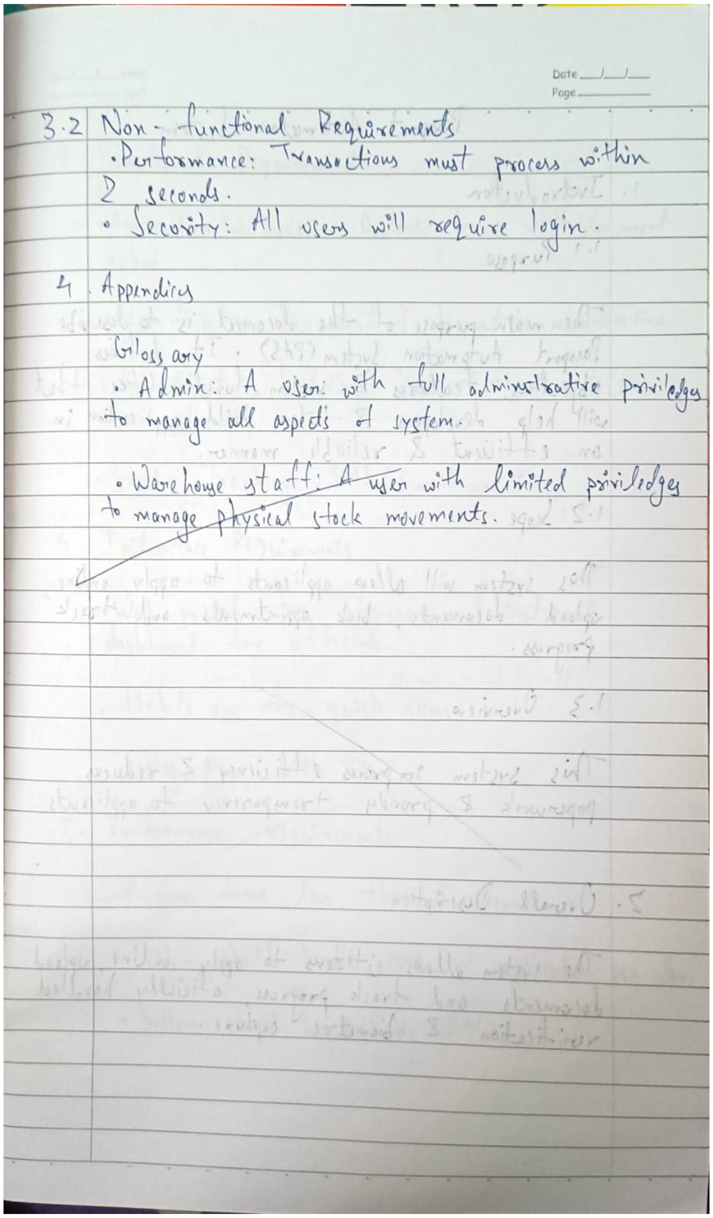
- Technology: Must use a modern web framework
- Security: Must use HTTPS & data encryption.
- Performance: Must handle upto 50 concurrent users.

3. Specific Requirements

3.1 Functional Requirements

- Admins can add, search for and edit products.

- The system will automatically update stock levels & issue low stock alerts.



Date ___/___/___
Page ______

3.2 Non-functional Requirements

· Performance: Transactions must process within 2 seconds.

· Security: All users will require login.

4. Appendics

Gloss ary

· Admin: A user with full administrative priviledges to manage all aspects of system.

· Warehouse staff: A user with limited privildges to manage physical stock movements.

Class Diagram

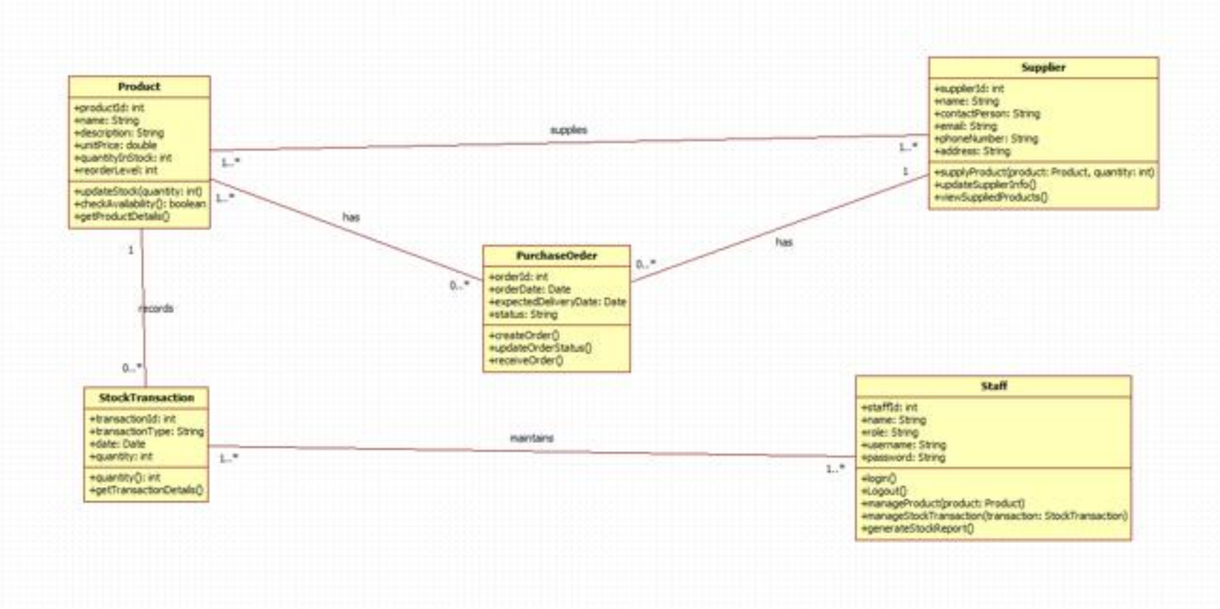


Figure 4.1: Class Diagram for Stock Maintenance System

This inventory management system tracks products supplied by various suppliers and maintained by staff members through purchase orders and stock transactions. Each supplier provides multiple products, and purchase orders are created to record order dates, expected delivery dates, quantities, and order status. When stock arrives or changes occur, stock transactions log the product, quantity, and transaction type to keep inventory accurate. Staff members manage the creation and updating of purchase orders, record stock transactions, and generate reports, while products maintain their own pricing, availability, and reorder information. Together, these components ensure smooth purchasing, stock maintenance, and supplier coordination.

State Diagram

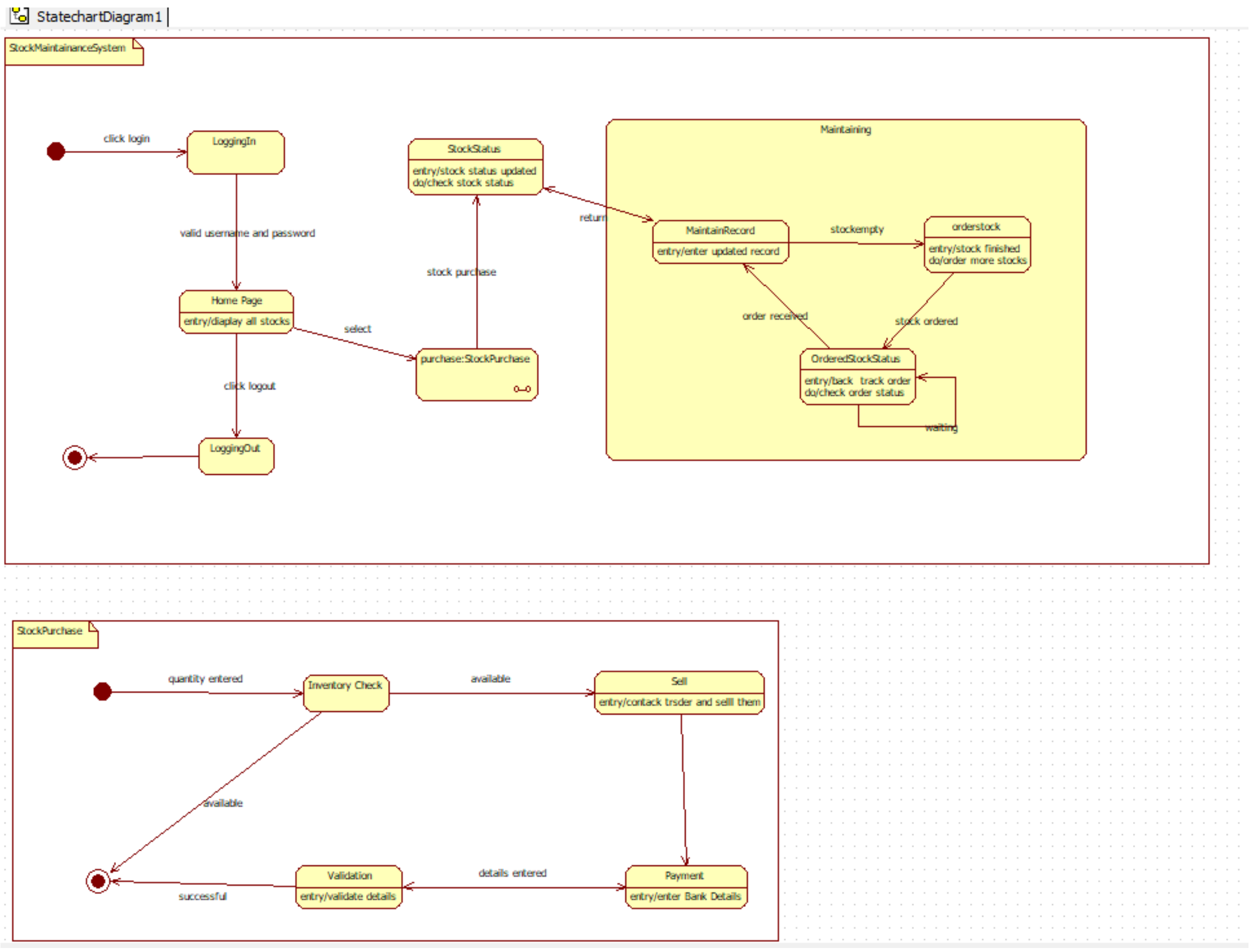


Figure 4.2: State Diagram for Stock Maintenance System

The stock maintenance process begins when a user logs into the system and is taken to the home page, where they can view all available stock items and choose to perform a stock purchase. When a stock purchase is initiated, the system checks current stock status and moves into the maintenance workflow, where records are updated, shortages are detected, and new stock orders are placed if inventory is low. The system then tracks the status of ordered stock until it arrives, updating records accordingly. Meanwhile, the stock purchase subprocess validates the requested quantity, determines whether sufficient stock is available, and either proceeds to sell the items and process payment or terminates the request if stock is insufficient. The overall process concludes when the user logs out or the purchase transaction successfully completes.

Use Case Diagram

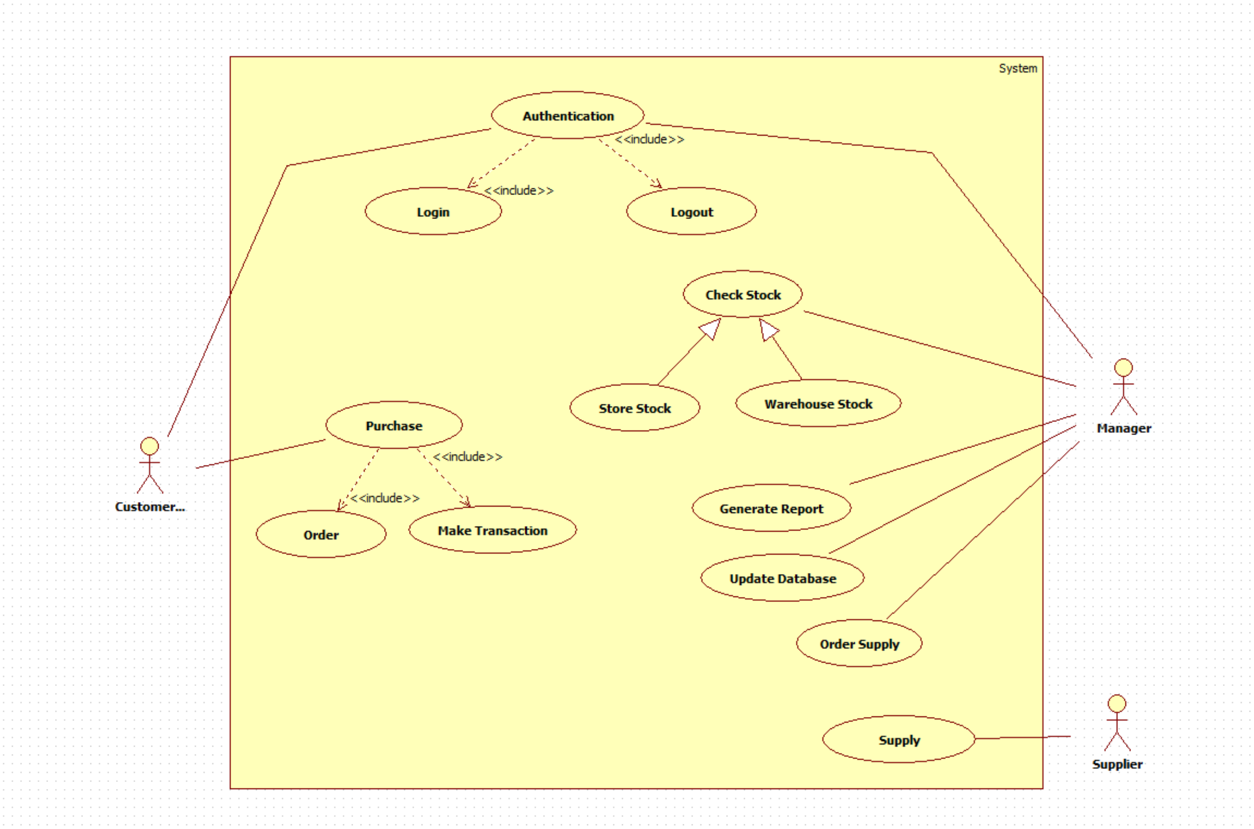


Figure 4.3: Use Case Diagram for Stock Maintenance System

In this stock management system, customers can purchase items by placing orders and making transactions, while the system ensures that user authentication through login and logout is always included. Managers interact with the system to check stock levels, whether in the store or the warehouse, and can generate reports or update the database based on stock conditions. When stock is low, the manager can initiate a supply order from suppliers, who then provide the required items to the system. Throughout the process, checking and managing stock is central, supporting smooth purchasing, inventory updates, and supply coordination.

Sequence Diagram

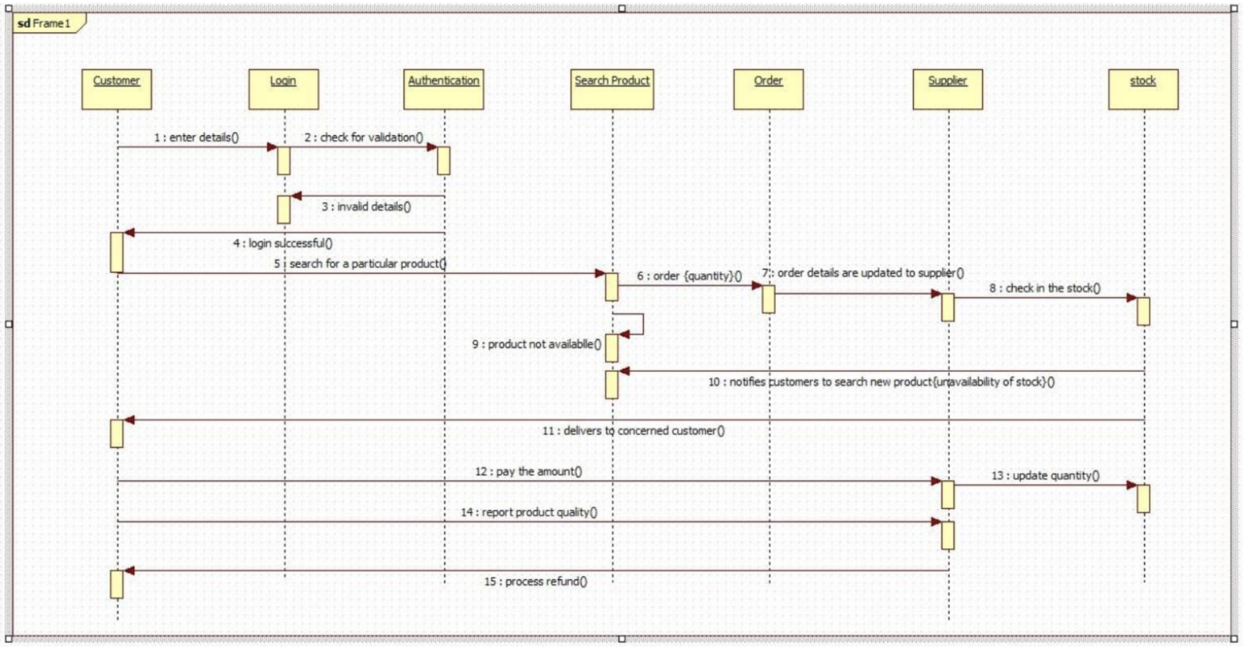

sd Frame1
Customer
Login
Authentication
Search Product
Order
Supplier
stock
1 : enter details()
2 : check for validation()
3 : invalid details()
4 : login successful()
5 : search for a particular product()
6 : order {quantity}()
7 : order details are updated to supplier()
8 : check in the stock()
9 : product not availablle()
10 : notifies customers to search new product{unavailability of stock}()
11 : delivers to concerned customer()
12 : pay the amount()
13 : update quantity()
14 : report product quality()
15 : process refund()

Figure 4.4: Sequence Diagram for Stock Maintenance System

In this sequence, a customer begins by entering login details, which the login system forwards to the authentication module for validation; if the details are invalid, the customer is notified, otherwise login succeeds. The customer then searches for a specific product, and the system initiates an order request with the desired quantity. The order information is sent to the supplier, who checks product availability in stock. If the product is unavailable, the supplier returns a notification prompting the customer to search for a new product. When the product is available, the supplier updates stock levels, processes the order, and delivers the product to the customer. The customer then pays for the product, reports product quality if needed, and the system may process a refund if a problem is reported, completing the transaction flow.

Activity Diagram

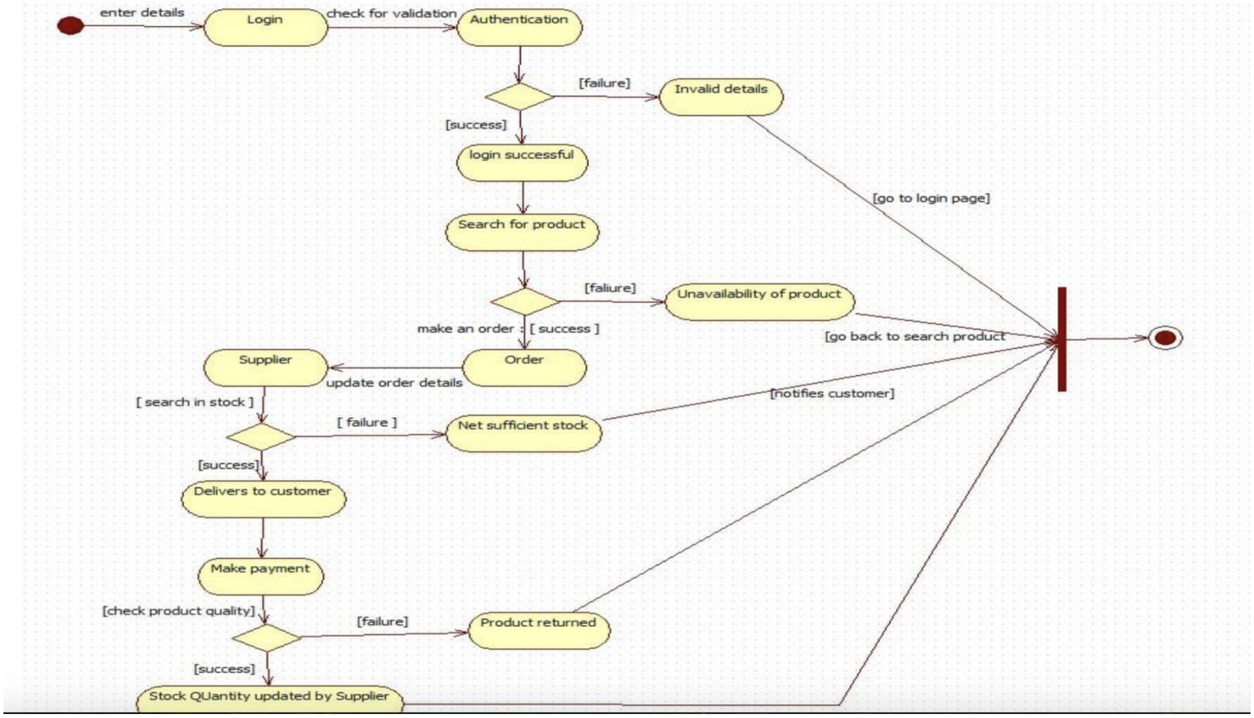


Figure 4.5: Activity Diagram for Stock Maintenance System

The activity begins when a customer enters their login details, which are validated by the authentication process; if the details are invalid, the flow ends by redirecting the user back to the login page. When login is successful, the customer searches for a product, and the system checks its availability; if the product is unavailable, the user is prompted to search again. If available, the customer places an order, and the supplier checks stock levels to confirm whether sufficient quantity exists. If stock is inadequate, the system notifies the customer and the process ends. When stock is sufficient, the supplier updates order details, delivers the product, and the customer makes payment. After purchase, product quality is checked; if the quality fails, the product is returned, otherwise the supplier updates stock quantity and the flow completes successfully.

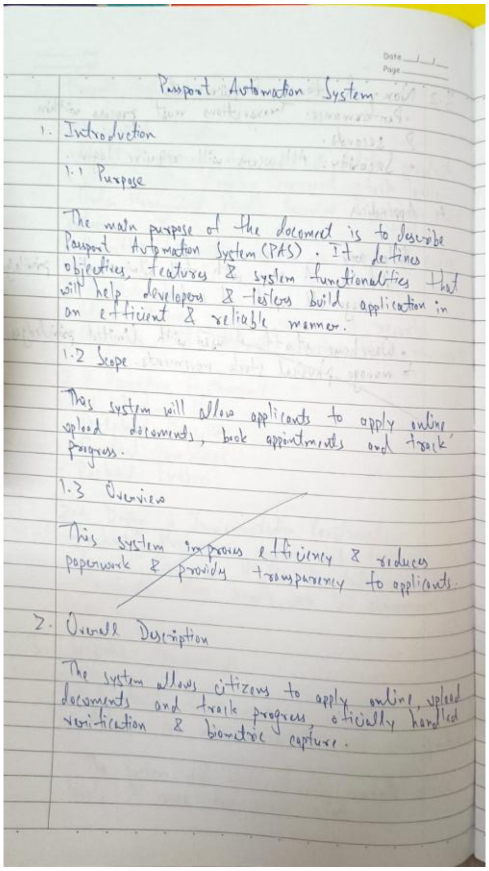
5. Passport Automation System

Problem Statement

The process of passport application and issuance in many regions involves several manual steps such as filling out physical forms, submitting documents in person, waiting in long queues, and undergoing slow verification processes. These traditional methods often lead to delays, inconsistencies in data, missing documents, difficulty in tracking application status, and a lack of transparency for applicants. To address these challenges and improve the efficiency of passport-related services, a computerized Passport Automation System is required.

The Passport Automation System must automate the major functions involved in applying for, processing, and issuing passports. Applicants should be able to submit their applications online, upload required documents, schedule appointments, and track the status of their application at each stage. The system must validate applicant information, verify supporting documents, and provide alerts or notifications if corrections or additional documents are needed. It must support various stages of processing, including police verification, document scanning, fee payment, background checks, and final approval.

SRS-Software Requirements Specification



Date __/__/__
Page ____

Passport Automation System

1. Introduction

1.1 Purpose

The main purpose of the document is to describe Passport Automation System (PAS). It defines objectives, features & system functionalities that will help developers & testers build application in an efficient & reliable manner.

1.2 Scope

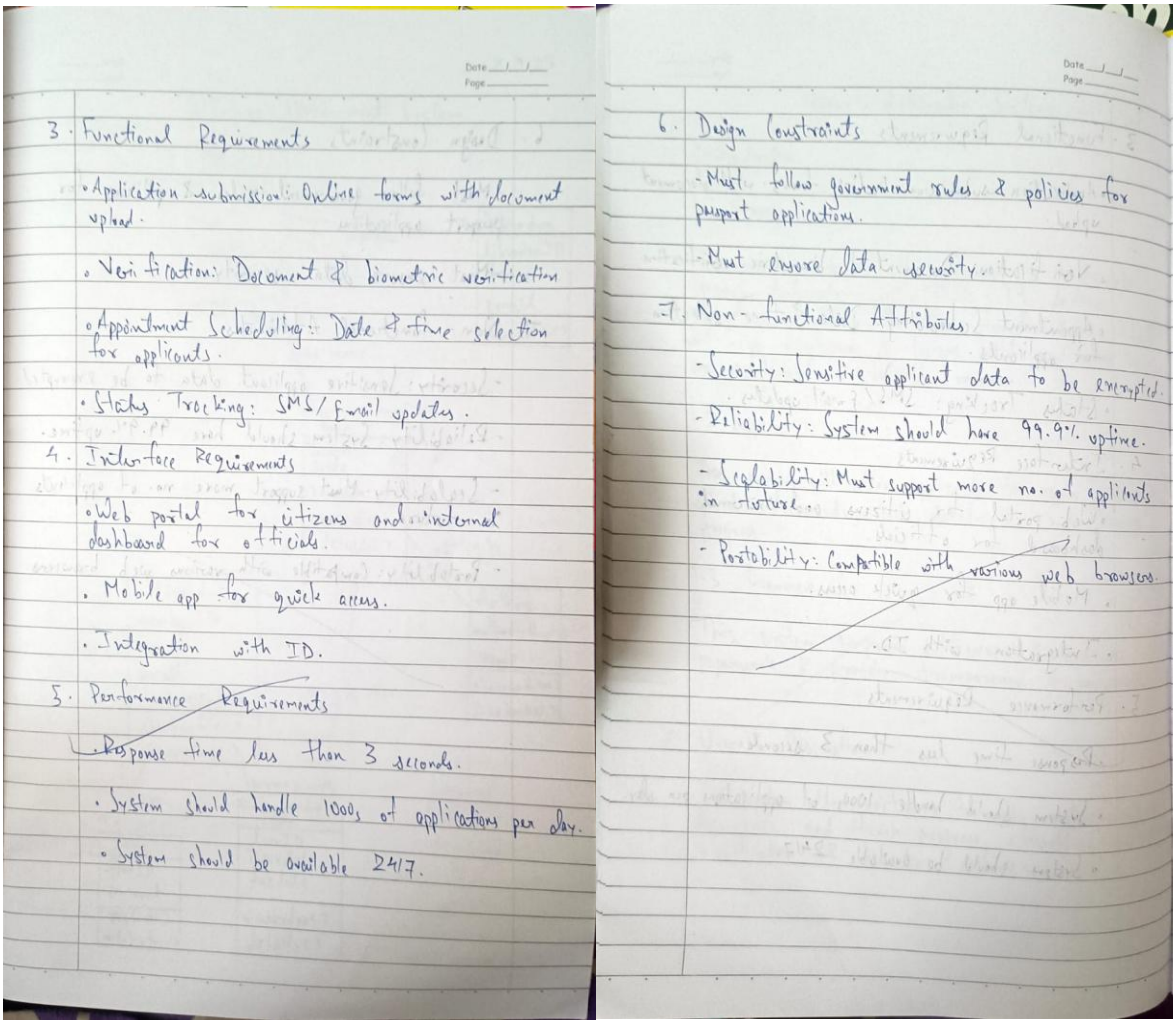
This system will allow applicants to apply online upload documents, book appointments and track progress.

1.3 Overview

This system improves efficiency & reduces paperwork & provides transparency to applicants.

2. Overall Description

The system allows citizens to apply online, upload documents and track progress, oficially handled verification & biometric capture.



Date __/__/__
Page ____

3. Functional Requirements

- Application submission: Online forms with document upload.
- Verification: Document & biometric verification
- Appointment Scheduling: Date & time selection for applicants.
- Status Tracking: SMS/Email updates.

4. Interface Requirements

- Web portal for citizens and internal dashboard for officials.
- Mobile app for quick access.
- Integration with ID.

5. Performance Requirements

- Response time less than 3 seconds.
- System should handle 1000s of applications per day.
- System should be available 24/7.

Date __/__/__
Page ____

6. Design Constraints

- Must follow government rules & policies for passport applications.
- Must ensure data security.

7. Non-functional Attributes

- Security: Sensitive applicant data to be encrypted.
- Reliability: System should have 99.9% uptime.
- Scalability: Must support more no. of applicants in future.
- Portability: Compatible with various web browsers.

Class Diagram

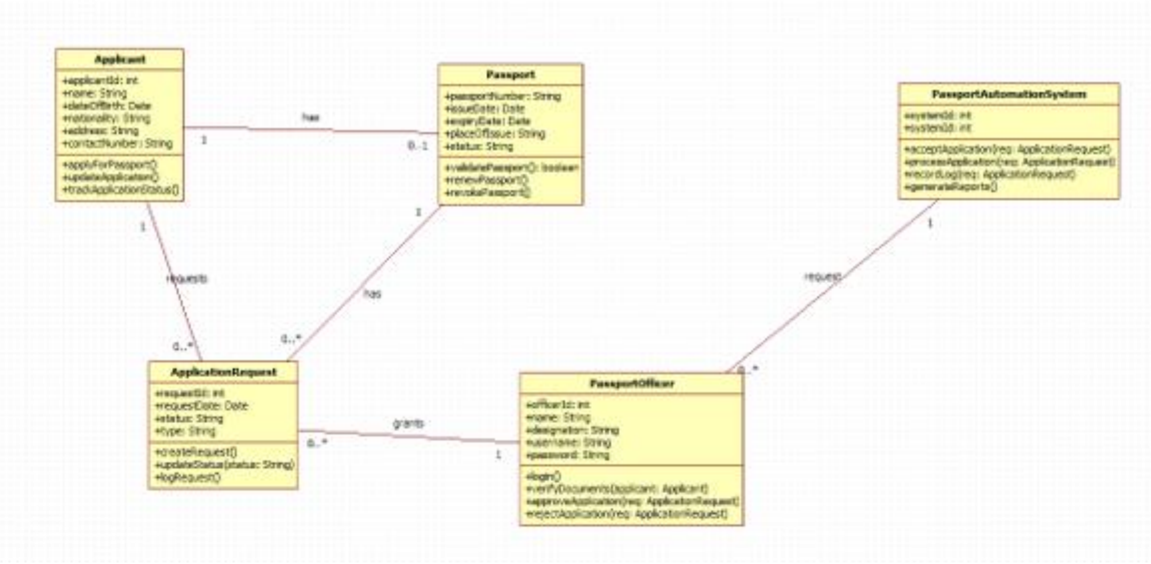


Figure 5.1: Class Diagram for Passport Automation System

This passport application system manages applicants who submit passport requests that are processed by passport officers through the passport automation system. Each applicant may already have an existing passport or apply for a new one by generating an application request containing details such as request type, submission date, and status. The passport officer reviews the application, verifies documents, approves or rejects the request, and updates the status accordingly. Approved requests result in the issuance of a passport linked to the applicant. Throughout the process, the passport automation system coordinates application submissions, retrieves applicant and passport data, forwards requests to officers, and returns confirmation or rejection results, ensuring a smooth and organized passport issuance workflow.

State Diagram

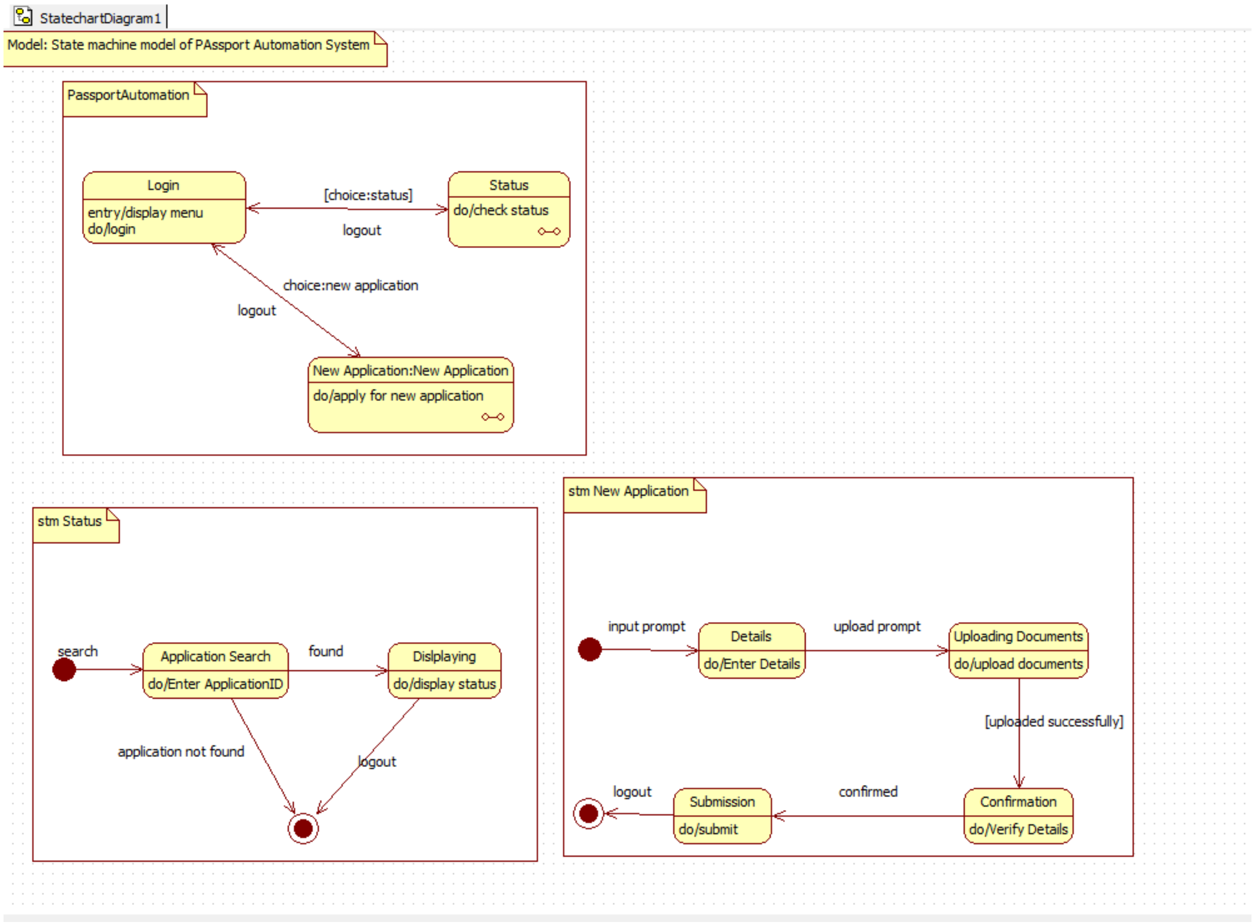


Figure 5.2: State Diagram for Passport Automation System

The passport automation system begins with the user logging in, after which they can either check the status of an existing application or start a new one. If the status option is chosen, the system enters the status state machine, where the user searches by application ID, and the system either displays the current status or ends the process if the application is not found. If the user selects to submit a new application, the system moves into the new-application workflow, prompting the user to enter personal details, upload required documents, and submit the form. The system then verifies the submitted information and confirms successful completion. At any point, the user can log out, returning the system to the initial login menu and resetting the workflow.

Use Case Diagram

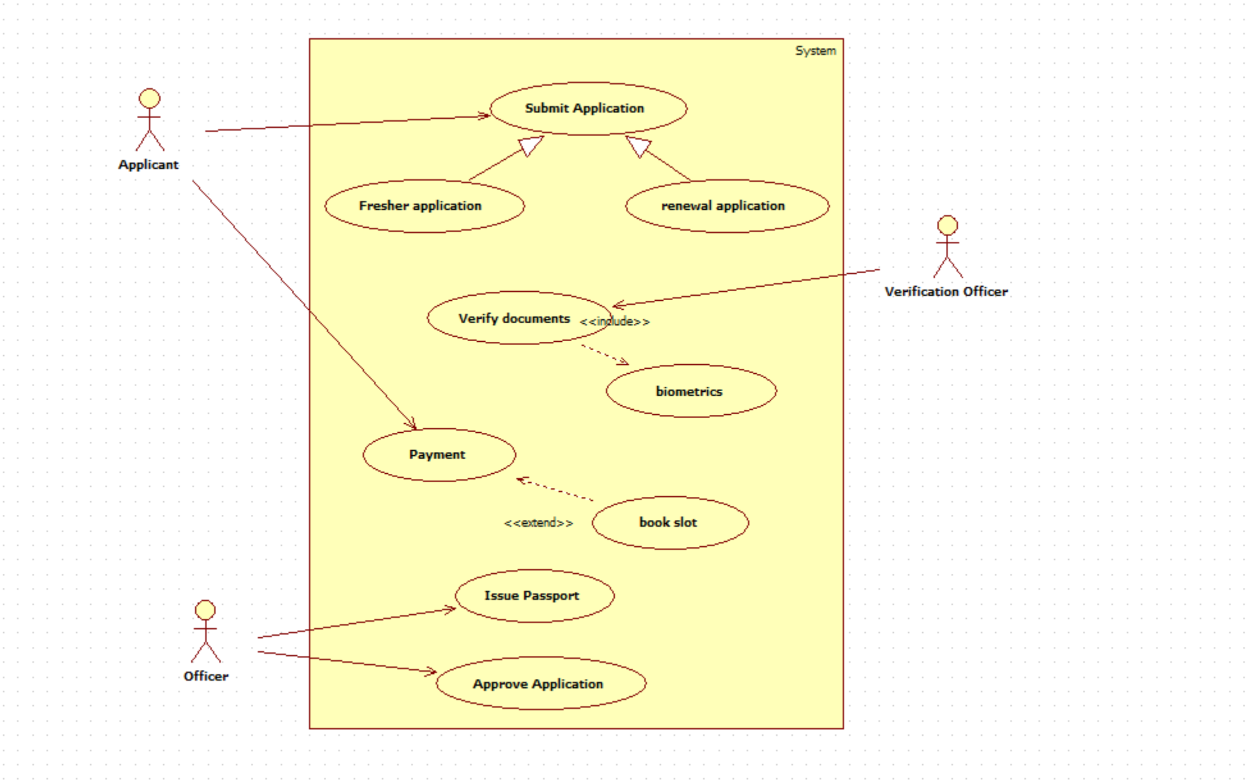


Figure 5.3: Use Case Diagram for Passport Automation System

This passport application system allows an applicant to submit either a fresh application or a renewal request, after which the system proceeds to verify the submitted documents with the involvement of a verification officer. As part of the verification process, the applicant also completes biometric procedures before selecting and booking an appointment slot. Once verification is completed, the applicant makes the necessary payment, and the system forwards the case to an officer who approves the application and issues the passport. Throughout the process, document verification is an essential included step, while slot booking extends the payment workflow to ensure proper scheduling and processing.

Sequence Diagram

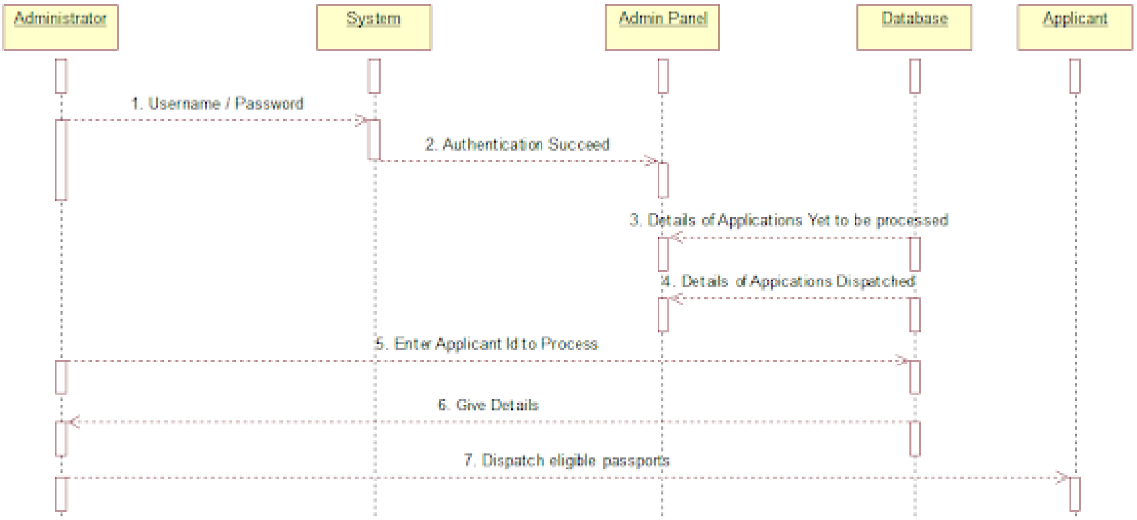

Administrator
System
Admin Panel
Database
Applicant
1. Username / Password
2. Authentication Succeed
3. Details of Applications Yet to be processed
4. Details of Appications Dispatched
5. Enter Applicant Id to Process
6. Give Details
7. Dispatch eligible passports

Figure 5.4: Sequence Diagram for Passport Automation System

In this passport dispatch workflow, an administrator begins by entering their username and password, which the system validates before confirming successful authentication. Once logged in, the system retrieves from the admin panel the list of passport applications that are still pending and those already dispatched, using the database as the data source. The administrator then selects an applicant ID to process, after which the admin panel sends the applicant's details back through the system. Finally, eligible passports are dispatched to the respective applicants, completing the sequence.

Activity Diagram

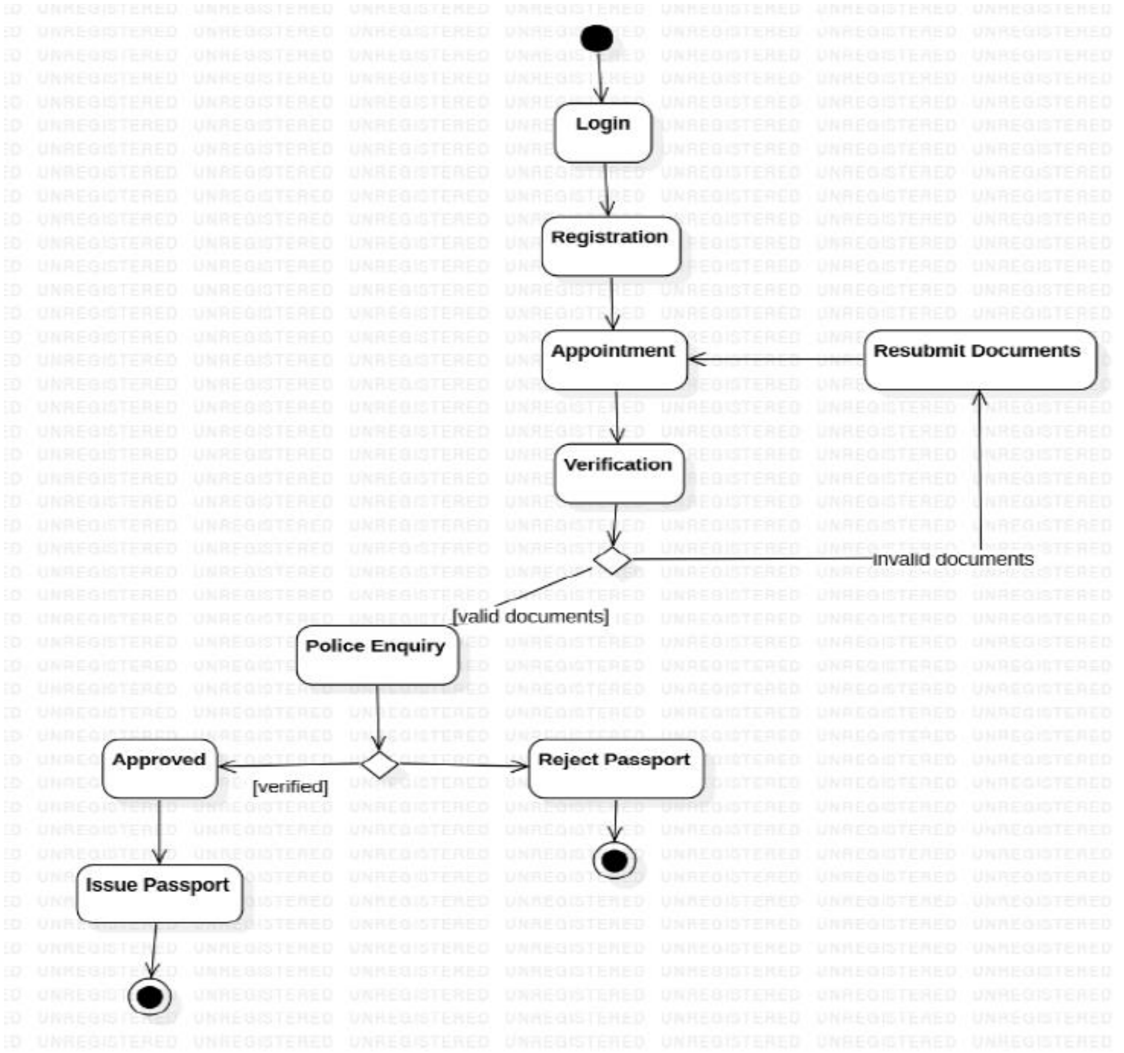


Figure 5.5: Activity Diagram for Passport Automation System

This passport approval process begins when the applicant logs into the system and completes the registration step, after which they schedule an appointment for document verification. During verification, the submitted documents are checked; if they are invalid, the applicant is asked to resubmit the documents and return to the appointment stage. If the documents are valid, the process proceeds to a police enquiry for background verification. Based on the police verification report, the application is either approved or rejected. Approved applications move to the final step where the passport is issued, while rejected applications terminate the process.